3-14-2014

Performance Characterization, Development, and Application of Artificial Potential Function Guidance Methods

Heather M. Stickney

Performance Characterization, Development, and Application of Artificial Potential Function Guidance Methods

THESIS

Heather M. Stickney, Capt, USAF

AFIT-ENY-14-M-44

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-14-M-44

Performance Characterization, Development, and Application of Artificial Potential Function Guidance Methods

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Aeronautical Engineering

Heather M. Stickney, BSAE

Capt, USAF

March 2014

AFIT-ENY-14-M-44

Performance Characterization, Development, and Application of Artificial Potential Function Guidance Methods

Heather M. Stickney, BSAE
Capt, USAF

Approved:

//signed//	12 March 2014
Lt Col Jeremy S. Agte, PhD, (Chairman)	date
//signed//	12 March 2014
Richard G. Cobb, PhD (Member)	date
//signed//	12 March 2014
David R. Jacques, PhD (Member)	date

Abstract

The primary objective of this research was to examine artificial potential function (APF) guidance performance when applied to systems with limited control authority in a dynamic environment and then to use the findings to develop a hybrid guidance to improve algorithm convergence and computational cost. Performance with respect to both computation time and cost was improved by hybridizing the APF approach with receding horizon optimal control planning. Results showed that for the hybrid algorithm, computation time was improved from the optimal control solution while improving the convergence and cost from the baseline APF solution. While the hybrid method greatly improved performance for a saturated system in dynamic environment, this was limited to a fully actuated system. When applied with indirect control, performance was improved, but did not converge. Based on this initial data, the hybrid approach shows promise in regard to implementation within a real-time guidance scheme, however, there is still work to be done before it will be fully effective. The secondary objective of this research was to determine what classes of problems are well-suited to APFs or APF-hybrids. The data suggests that APFs and the hybrid algorithm proposed are best applied to fully actuated systems. Additionally, if external dynamics or substantial saturation exist, APF guidance performs better when supplemented with an alternative method.

Acknowledgements

I sincerely thank Lt Col Jeremy Agte for his guidance, patience, and instruction throughout this thesis, as well as the many hours spent editing this document. I also thank Dr. Richard Cobb for his help during Lt Col Agte's absence. I am grateful for the opportunity to have worked with you both and have learned a lot throughout the course of this research.

Heather M. Stickney

Table of Contents

Page

List of Figures

List of Tables

List of Symbols

List of Abbreviations

Abbreviation		Page
APF	artificial potential function	1
OCP	optimal control problem	4
EMFF	electromagnetic formation flying	12
AAPF	adaptive artificial potential function	13
RH	receding horizon	47
CTG	cost-to-go	48

Performance Characterization, Development, and Application of Artificial Potential Function Guidance Methods

I. Introduction

1.1 Motivation and Background

As autonomous systems become more prevalent, there is an increased need for precision real-time guidance solutions. Current autonomous aircraft guidance systems typically exert control via way-point navigation or are designed to loiter about a specified point. Waypoint navigation traditionally employs simple feedback loops to eliminate track error and maintain altitude and/or airspeed schedules. A loiter system may simply compute leg times and turn radius to indefinitely suspend at a given location. While these types of guidance systems are abundant, they do not normally account for a performance index in a more global sense. While the algorithm may internally minimize track or altitude error with or without minimizing control effort, it usually does not attempt to minimize broader performance indexes such as transit time or fuel consumption. This is largely because real-time optimal guidance solutions that are robust to changes within a dynamic environment still represent a challenge to solve due to their complexity and computational requirements.

With this in mind, the need for high-performing, computationally efficient guidance algorithms is evident. Various solutions have been proposed and analyzed by the guidance and control community to include pseudospectral optimization methods, neural networking, evolutionary algorithms, and artificial potential functions. While there has been significant research in these areas, a research gap is evident in the study of artificial potential functions (APF). APFs have been successfully applied in the fields of robotics [1] and satellite rendezvous and proximity operations [2, 3], but little work has been done to improve APF-based algorithms with respect to perfor-

mance indexes or robustness. Additionally, little work has been done to assess the utility of APFs with respect to systems that are not fully actuated[1] and/or have limited control authority.

1.2 Artificial Potential Functions

Artificial potential functions were created to serve as reactionary trajectory planning algorithms. They are scalar-valued functions that result from the superposition of attractive and repulsive potential fields [4]. These attractive and repulsive potential fields are typically Gaussian or harmonic functions modeled to drive the system to a desired set of states as well as avoid obstacles. The attractive field is designed to create a global minimum at the goal state. The repulsive field consists of areas of high potential modeled at the obstacles. If the combined potential function can be designed without local minima then the objective is guaranteed to be reached while also avoiding collisions [5]. Figure 1.1 shows the gradients of an example potential field.

The appeal of APF guidance is the simplicity of implementation and computational efficiency. The guidance solution results from following the negative gradient of the potential field at all times. Since the potential field is designed as a differentiable analytic expression, the gradient computation is a simple evaluation of an analytic expression.

The form of APF implementation is typically chosen such that it possesses certain characteristics. These characteristics include a unique global minimum and absence of local minima. A unique global minimum can usually be guaranteed, but preventing local minima is quite difficult. The existence of local minima in the vector field allows the possibility for the vehicle to become trapped before the goal state has been reached. Unfortunately, in a dynamic environment it is often impossible

[1]Fully actuated refers to a system in which direct and instantaneous control of the vehicle's acceleration is possible.

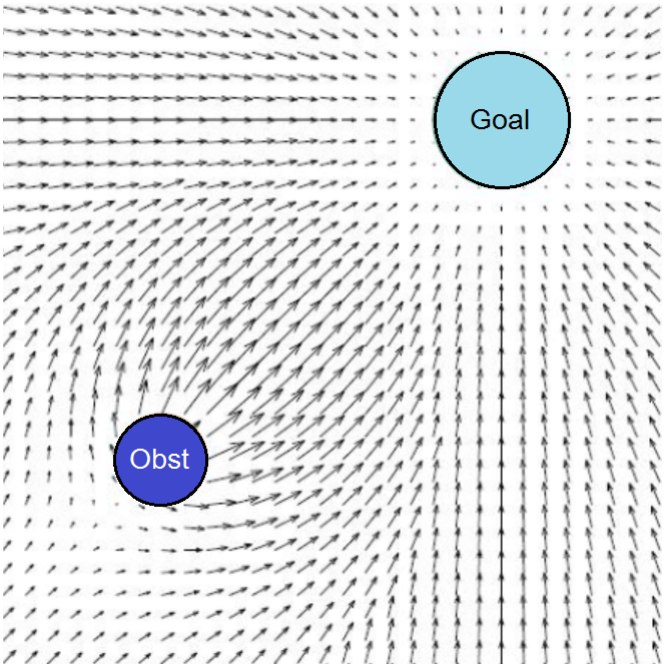

Figure 1.1: Artificial potential function illustration [4]

to guarantee that the potential space is devoid of local minima[2]. Instead, it is common practice with traditional APF guidance to include a perturbation algorithm to move the vehicle away from a local minimum (if trapped) and ensure the goal state is reached. Another local minima preventative measure includes grouping nearby obstacles to form a single larger obstacle. This practice is also used to address issues with arbitrarily shaped obstacles.

Assuming any local minima issues have been mitigated, converging to a global minimum is only possible if the vehicle can actually follow the gradient of the potential field. It is conceivable that the available control power of the vehicle can be less than that required by the APF. In this case, convergence to the global minimum is still possible, but no longer guaranteed because the path of steepest descent is no longer being followed. By the same logic, a collision free path cannot be guaranteed.

[2]The issue of local minima is a limitation of the gradient descent method and is not unique to artificial potential functions.

Difficulties can also exist when the force required by the APF to follow the negative gradient of the potential field does not directly translate to a control. In satellite dynamics, the required force can be directly achieved by firing thrusters. However, if the control is an actuator, translation of the required force to a control is often not straightforward.

This thesis aims to address the issues of controllability and indirect force actuation. Additionally, the impact to the APF trajectory due to external dynamics will be examined.

1.3 Optimization

Optimization within the guidance framework is desirable when a performance index is important. This includes minimum fuel, minimum transit time, maximum range, etc. Optimal control theory is derived from the calculus of variations which consists of maximizing or minimizing functionals.

Optimal control problems (OCP) must be solved at a high refresh rate if they are to be used in real-time guidance in a dynamic environment. Often, the computational burden and processing limitations of finding the OCP solution prevent it from being generated fast enough for real-time guidance applications.

Artificial potential functions have shown promise in guidance solutions due to their computational efficiency since the solution results from evaluating an analytic expression. Very high sample rates can be used with APF guidance, however, APF solutions are inherently suboptimal and lack robustness.

1.4 Thesis Objectives and Methodology

The goal of this research is to examine APF performance when applied to systems with limited control authority and external dynamics and use the findings to develop a hybrid guidance algorithm based on an appropriate measure of tradeoff between computational efficiency and optimality.

To highlight the benefits and shortcomings of each method, a simple two-dimensional case study will be developed and analyzed to compare and contrast APFs with optimal control solutions.

The secondary objective of this research is to determine what classes of problems are well-suited to APFs or APF-hybrids. For example, satellite rendezvous with Clohessy-Wiltshire[3] dynamics are well-suited to APF guidance because gravity is the only driving force and the control is fully actuated via thrusters. It is to be determined if APFs can be effectively implemented for more complex problems such as terrestrial navigation where external forces have significant impact on vehicle dynamics.

1.5 Thesis Outline

A literature review containing background information on artificial potential functions, optimal control theory, and relevant works will be detailed in Chapter 2. In Chapter 3, simple case studies will be used to characterize the performance of APF guidance methods. Comparisons will also be made to optimally planned trajectories. Chapter 4 will detail the development and application of a hybrid guidance algorithm that combines receding horizon optimal control planning with artificial potential function guidance. Finally, a summary of the work completed will along with conclusions and recommendations for future work will be presented in Chapter 5.

[3]The Clohessy-Wiltshire equations represent a linearized model for the rendezvous of a chaser satellite with a target satellite.

II. Literature Review

A brief history of artificial potential functions and their applications is provided as well as an overview of optimal control theory as it relates to this research.

2.1 Artificial Potential Functions

Artificial potential functions provide the capability of real-time guidance in a dynamic environment. Previous research on their development and application has included satellite rendezvous and proximity operations [3,6], terrestrial robot navigation [5,7], unmanned underwater vehicle navigation [8,9], and cooperative unmanned aerial vehicle control [10, 11]. Artificial potential functions are appealing due to the computational efficiency of the gradient descent technique.

2.1.1 Types of Artificial Potential Functions. As mentioned in Chapter 1, the defined potential fields are differentiable thus an analytic expression for the gradient exists. The exact form of the gradient is determined by the type of APF implemented. Historically, attractive potential fields, ϕ_a, have been modeled as Gaussian (Eq. 3.10), harmonic (Eq. 2.2), and quadratic functions (Eq. 2.3) [3, 6, 12].

$$\phi_a(x,y) = -\lambda \exp\left[-\left(\frac{x-x_o}{\alpha} + \frac{y-y_o}{\beta}\right)\right] \tag{2.1}$$

$$\phi_a(x,y) = -U\left(x\cos\alpha + y\sin\alpha\right) + \frac{\lambda}{2\pi}\log\left(\left(x-x_o\right)^2 + \left(y-y_o\right)^2\right)^{1/2} \tag{2.2}$$

$$\phi_a(\vec{x}) = \frac{1}{2}\left(\vec{x}-x_f\right)^T Q\left(\vec{x}-x_f\right) \tag{2.3}$$

Forms of repulsive potentials fields, ϕ_r, have included exponentials, Eq. (2.4) and simple distance functions, Eq. (2.5) [6].

$$\phi_r(\vec{x}) = \lambda \exp\left[-\frac{(\vec{x}-\vec{x}_o)^T N (\vec{x}-\vec{x}_o)}{\sigma}\right] \tag{2.4}$$

$$\phi_r(x,y) = \frac{\lambda}{\left((x-x_o)^2 + (y-y_o)^2\right)^{1/2}} \tag{2.5}$$

Each of the formulations have what are normally referred to as shaping parameters (Q, α, β, λ, N). These parameters are usually constant and empirically determined, but can be modeled as functions of time.

2.1.2 Drawbacks of Artificial Potential Functions. Inherent issues with APFs include local minima within the potential space and oscillatory movement [4]. Local minima can be avoided by including perturbation algorithms in the guidance subroutine, although it is normally more desirable (and more difficult) to design the potential field such that these local minima do not exist. Oscillatory movement is typically addressed by setting an APF control threshold. This threshold simply prevents the APF guidance algorithm from updating the control command until it exceeds some minimum value. The control threshold is usually set empirically.

Additionally, APF trajectories are often abrupt and may violate rate limit constraints on systems such as aircraft. If rate limit constraints are imposed, care must be taken to adjust parameter gains in order to ensure the system is still guaranteed to be collision free. Ahsun asserted that this saturation can sometimes be avoided via proper scaling of the potential field in areas that may cause large position errors [6]. Saturation can also be prevented in certain instances by including additional repulsive potentials on the controls [13].

When concerned with performance measures, an APF formulation can be a poor choice as the lack of feedback makes it inherently suboptimal and the influence of system dynamics is often minimal or non-existent. However, APFs remain desirable due

to their computational efficiency. This research will focus on methods of improving APF optimality and convergence.

2.2 Similar Methods

Methods such as stream functions and navigation functions which very similar to APFs have been previously investigated. Both were explored as ways of mitigating local minima and have their own unique pros and cons. Stream functions represent a special class of harmonic functions and navigation functions are a special class of artificial potential functions.

2.2.1 Stream Functions. A similar technique using stream functions was investigated by Waydo [5]. Instead of following the negative gradient of a potential field, the trajectory follows streamlines using stream functions that model inviscid, incompressible, irrotational fluid flow. In many cases, the obstacle is modeled as a source, the goal is modeled as a sink, and a uniform flow field is superimposed to provide a propulsive force toward the goal as depicted in figure 2.1.

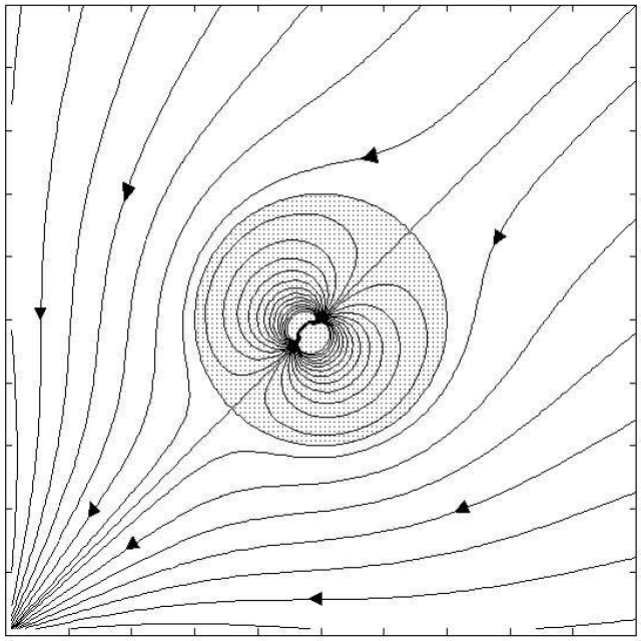

Figure 2.1: Stream function obstacle avoidance [5]

Equation (2.6) provides an example stream function which consists of a source/sink in uniform flow [12].

$$\phi = -U\left(x\cos\,\alpha + y\sin\,\alpha\right) + \frac{\lambda}{2\pi}\log\,r \tag{2.6}$$

This method differs from traditional APFs in that it is free of local minima and provides much smoother trajectories. The latter is beneficial for systems with rate constraints. The main disadvantage of stream functions is that they are limited to two-dimensional applications due to the nature of their formulation.

2.2.2 Navigation Functions. Another formulation that attempts to mitigate the problem of local minima is that of navigation functions. They are designed such that, with the exception of the global minimum at the goal, all points where the gradient is zero are unstable saddle points [14]. The obvious benefit of this type of potential field is that the trajectory generated is guaranteed to converge at the desired goal with zero possibility of being trapped in a local minima. However, the creation of a function with such characteristics has proven to be very difficult and the functions themselves are typically very complex. As a result, significant computational power much be used to develop the functions outside of the guidance algorithm [9]. Equation (2.7) represents an example of a navigation function [11].

$$\phi\left(q\right) = \frac{\gamma + f}{\left(\left(\gamma + f\right)^{k} + H_{nh}\cdot G\cdot\beta_{0}\right)^{1/k}} \tag{2.7}$$

where:

$\gamma = \|q - q_d\|^2$

q and q_d are vectors of states and desired states, respectively

k is a scale factor greater than zero

f is a function of G that ensures ϕ is always positive

H is responsible for aligning the trajectory with the desired orientation

β_0 is an obstacle that bounds the workspace

G is a proximity function related to possible collisions

This equation can be better visualized by examining figure 2.2 which is an example of a dipolar[1] navigation function.

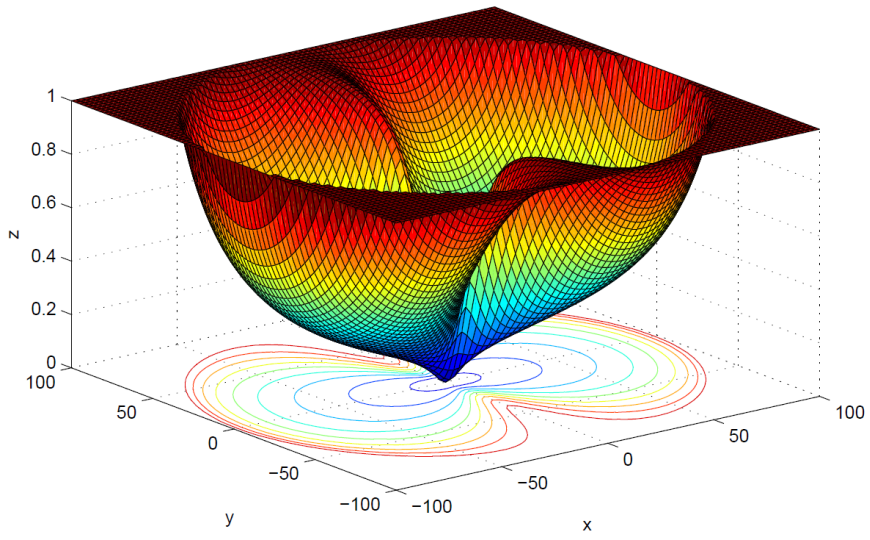

Figure 2.2: Dipolar navigation function [11]

Because of the complex formulation and substantial offline computational burden, navigation functions were not further explored in this research.

[1] A dipolar navigation function is a special navigation function that allows a vehicle to arrive at its destination with a specific orientation [11].

2.3 Adaptive and Evolutionary Artificial Potential Functions

Researchers in the APF community widely recognize the need to improve performance and robustness of these algorithm. As such, several have proposed alternative methods of APF implementation in order to do so. These methods involve incorporating feedback into the APF guidance scheme as well as using what have been termed adaptive and evolutionary APFs.

2.3.1 Basic Feedback. Healey investigated APFs for use in path tracking for an underwater autonomous vehicle operating at constant speed and depth [8]. The goal was to minimize deviation from a desired straight-line path when currents were present while also avoiding obstacles on and near the path. Healey began by employing a feedback controller to drive cross track error to zero. Once the feedback controller successfully compensated for the water current, both attractive and repulsive potentials were implemented within an outer loop for path planning purposes. The attractive potential encouraged the vehicle to follow a straight line path between way points while the repulsive potential ensured a collision free path. Recognizing that the various APF parameters could be tuned to improve performance, Healey noted the possibility for optimized path planning.

2.3.2 Evolutionary APFs. The use of evolutionary, or genetic, algorithms have great appeal in optimization because they can sometimes outperform their gradient based counterparts. Namely, they tend to be less sensitive to local minima in the solution space. Genetic algorithms work by imitating the natural selection process to evolve the best candidates to the optimal solution. Recognizing their utility, work has been done to use genetic algorithms in conjunction with APFs to optimize the shaping parameters with respect to a performance index.

Vadakkepat et al. developed a real-time path planning technique for robots that used a genetic algorithm to obtain an optimal potential field by tuning the repulsive potential parameters associated with each obstacle [7]. The research successfully

created simple, tuned APFs that resulted from minimizing a performance index. The authors selected three cost functions with which to optimize the APF: minimum path length, goal-factor, and obstacle-factor. Goal-factor was minimized when the robot was at the goal and the obstacle-factor was minimized any time the robot did not collide with the obstacle. The optimal APF as determined by the genetic algorithm was then one that reached the goal, avoided obstacles, and did so with the minimum total path length. Additionally, to avoid local minima, the authors implemented an escape force within the algorithm when it was recognized that a local minima existed. Since the algorithm was a real-time guidance application, an optimal APF was determined at each point along the robot trajectory meaning the APF parameters were a function of time. This produced an algorithm that was effective for both stationary and moving obstacles and goals. However, the authors recognized that the trajectory obtained was a function of the type of APF chosen and suggested that performance could be further improved by not only optimizing the APF parameters, but also the type of APF implemented at each step.

2.3.3 Adaptive APF Techniques. While the evolutionary APF technique focused on a way of optimizing the repulsive potential to improve performance with respect to a cost function, others have recognized a marked increase in stability and robustness by embedding the system dynamics into the APF formulation [3, 6, 15].

Ahsun investigated Electromagnetic Formation Flying[2] (EMFF) and how to account for disturbance torques introduced by the Earth's magnetic field [6]. Because of the uncertain dynamics of EMFF, this research is of particular use when considering APF guidance in non-deterministic or stochastic environments. Ahsun's method used a Control Lyapunov Function[3] to generate a feedback control law within the APF guidance. This was done by choosing a form for the vehicle's acceleration vector such

[2]Electromagnetic Formation Flying is a technique that uses on-board electromagnets to control forces and torques between satellites in formation [6].

[3]A Control Lyapunov Function is a function used to determine if a system is stabilizable with feedback.

that asymptotic stability is guaranteed by Lyapunov theory. The advantage of this method are guarantees on system robustness.

With this method, the satellites' convergence rate tended to be slow as they approached very near to their goal. As a result, Ahsun introduced potential function shaping to change the shape of the gradient at the goal location. Adding the potential function shaping essentially added a second attractive potential with a steeper slope near the goal resulting in improved convergence time. The APF trajectories generated through Ahsun's method based on Lyapunov functions were then used as initial guess for non-linear optimization techniques.

Muñoz also recognized that APFs lack consideration of the system dynamics or a performance index [3]. He addressed this by first implementing a feedback controller to track the negative gradient of the APF for satellite rendezvous operations. He then attempted to improve APF performance by formulating an adaptive APF (AAPF) which used an update law to determine the attractive potential weighting parameters as a function of time. The repulsive potential weighting parameters were left unchanged. The weighting parameters were varied such that the difference between the negative gradient of the attractive potential and a desired velocity profile was minimized. The desired velocity profile was determined via the solution of a two point boundary value problem, so by matching the velocity profile, the dynamics were essentially embedded. Muñoz's method also suffered from slow convergence near the goal since it was approached asymptotically. As such, for satellite operations, he recommended that the guidance solution be split in into a close-range rendezvous phase in which the AAPF scheme is used and a final approach phase to reduce maneuvering time.

Current research by Fields has led to the development of a continuous control law within the APF framework [15]. This method allows for the dynamics of the system to be embedded within the APF guidance algorithm and uses them to generate a trajectory that is guaranteed to follow a path of decreasing potential. At any given

point, the vehicle trajectory may not point along the local negative gradient, but it is guaranteed not to increase in potential thus satisfying convergence and obstacle avoidance criteria. A limitation of the this approach is that the vehicle must have full control of it's acceleration vector. While this is of no issue to thrust-based controllers like those on satellites, vehicles with indirect control cannot benefit from the ingenuity of this method. Additionally, the continuous control method, like the other APF methods, is suboptimal as it does not track a performance index.

2.4 Optimal Control Theory

Since this research compares APF guidance solutions to corresponding optimal control solutions, it is important to briefly discuss the fundamentals of optimal control theory. Optimal control theory strives to find a control, $\vec{u}(t)$ such that a performance index is minimized while satisfying the constraints of the system dynamics and boundary conditions [16]. Optimal control theory applies to both continuous and discretized systems. Often, continuous systems are approximated by discretized systems, as is done in this research. This discussion on optimal control theory will be brief and limited to open final time problems due to the scope of this work. For open final time problems, Bryson defines the following pertinent equation, Eq. (2.8), as the relevant cost function [17].

$$J = \phi\left(\vec{x}\left(t_f\right), t_f\right) \tag{2.8}$$

Subject to:

$$\dot{\vec{x}} = f\left(\vec{x}\left(t\right), \vec{u}\left(t\right), t\right) \tag{2.9}$$

$$t_0,\ \vec{x}_0, \text{ and } \psi\left(\vec{x}_f, t_f\right) = 0 \tag{2.10}$$

The dynamic[4] and terminal[5] constraints given by Eqs. (2.9) and (2.10) can then be appended to the cost function to form what is termed the Lagrangian function, Eq. (2.11). It should be noted that ϕ is simply t_f for minimum time problems.

$$\bar{J} = \phi + \nu^T \psi + \int_{t_0}^{t_f} \vec{\lambda}^T(t) \left\{ f[\vec{x}(t), \vec{u}(t), t] - \dot{\vec{x}} \right\} dt \tag{2.11}$$

In Eq. (2.11), $\vec{\lambda}$ represents the vector of costates. From this point, the calculus of variations is used to obtain the first order necessary conditions for a candidate minimum. The reader is referred to Bryson [17] or comparable optimal control resources for the full derivation. For the purpose of this research, it will suffice to say that the first order necessary conditions are satisfied by the solution of the Eqs. (2.12)-(2.14).

$$\dot{\vec{x}}^*(t) = \frac{\partial H}{\partial \lambda} \left(\vec{x}^*(t), \vec{u}^*(t), \vec{\lambda}^*(t), t \right) \tag{2.12}$$

$$\dot{\vec{\lambda}}^*(t) = -\frac{\partial H}{\partial x} \left(\vec{x}^*(t), \vec{u}^*(t), \vec{\lambda}^*(t), t \right) \tag{2.13}$$

$$0 = \frac{\partial H}{\partial u} \left(\vec{x}^*(t), \vec{u}^*(t), \vec{\lambda}^*(t), t \right) \tag{2.14}$$

where:

$$H(t) = \vec{\lambda}^T(t) f[\vec{x}(t), \vec{u}(t), t] \tag{2.15}$$

The form of the Hamiltonian, H, presented in Eq. (2.15) is valid only for open final time problems. Equations (2.12)-(2.14) form the Euler-Lagrange equations and, along with the terminal conditions, must be satisfied at each point in time for a feasible solution to exist. Various methods for solving these equations exist to include forward and backward sequencing the state and co-state equations (Eqs. (2.12) and (2.13),

[4] A dynamic constraint is a constraint that is met by the satisfaction of the equations of motion.
[5] A terminal constraint is a constraint that is met by the satisfaction of the boundary conditions.

respectively). For this research, a more direct optimization approach was used by employing the sequential quadratic programming tool included in MATLAB®. This method solves nonlinear programming problems by minimizing a quadratic approximation of the Lagrangian function with linear approximations of the constraints [17].

2.5 Artificial Potential Functions and Optimal Trajectories

While Vadakkepat et al. used genetic algorithms to optimize APFs [7], Henshaw recognized a direct link between APFs and optimal control trajectories [18]. Henshaw proposed that optimal planning and APF guidance are related via receding horizon planning. Receding horizon planning is a method that optimizes a trajectory over a finite-time horizon of $(t_0, t+h)$ as opposed to (t_0, t_f). It is assumed that h is much smaller than $t_f - t_0$. The trajectory is then implemented for a time step, δ, that is smaller than h. As the trajectory is being followed, a new trajectory is planned for $(t+\delta, t+h+\delta)$. Henshaw pointed out that an APF can be thought of as a receding horizon problem where $h = 0$ and that optimal planning represents the case where $t+h = t_f$. With this assertion, he showed that APFs actually minimize a cost function of the form given by Eq. (2.16).

$$J\left[\vec{x}\left(t\right), t\right] = \int_{t}^{t+\delta t} \beta \sum_{j=1}^{n} R\left(dist\left(\vec{x}\left(t\right), c_j\right)\right) dt + E\left(\vec{x}\left(t+\delta t\right)\right) \tag{2.16}$$

Where:

$\beta \sum_{j=1}^{n} R\left(dist\left(\vec{x}\left(t\right), c_j\right)\right)$ is the repulsive potential

E is the attractive potential

From the limits of integration, it is obvious that the APF only minimizes over the next time step. It should be noted that Henshaw's use of APF guidance differs slightly from that presented so far in this research. As previously introduced, APFs were used

to determine a control whereas Henshaw's method uses APFs to generate waypoints. After noting the relationship among the trajectory planners, Henshaw conducted a case study to compare performance of the three algorithms. Each algorithm guided a robot through a 2D field of randomly generated obstacles and the respective costs and convergence times were recorded. As expected, the optimal trajectories generated had a much improved cost over the APF trajectory while the computation time was greatly reduced for the APF. The receding horizon planner produced costs much closer to the optimal solution while exhibiting convergence times much closer to APF guidance. In fact, for one obstacle configuration, the receding horizon planner actually produced a lower cost that the optimal trajectory. Henshaw attributed the results to the nonlinear optimization algorithm finding a local minima rather than the global minimum.

2.6 Summary

As evident from the summaries provided above, the author recognizes the need to improve APF performance while retaining computational efficiency. While various methods have been explored, to this author's knowledge, aside from Vadakkepat et al. [7] and Henshaw [18], little work has been done to directly implement APFs into an optimization scheme. This thesis aims to contribute to the APF and optimal control research community by attempting to do so by way of a hybrid algorithm that combines receding horizon and APF path planning.

Additionally, most prior research on APFs assumes adequate control power is available to perform a strict gradient descent. This was likely intentional since intuitively, neither obstacle avoidance nor goal capture can be guaranteed if a vehicle cannot execute the control required to follow the negative gradient. However, it is conceivable that for a vehicle with limited control authority and its corresponding dynamics, a feasible optimal control solution may still exist. In this case, there is obviously an attainable trajectory since the optimal control solution converged. One of the goals in the performance characterization portion of this research is to determine

how an APF can be manipulated such that the trajectory is driven as closely as possible to the optimal control solution.

III. Performance Characterization

This chapter details the methods used to characterize the performance of APFs in dynamic, control-limited environments. A case study with a simplistic concept yet difficult solution was selected. The test scenario to be proposed is a problem that scales well in difficulty to test potential guidance algorithms. As related to APF guidance, some of the difficulties of the case study include:

1. Indirect control, i.e. the control is not a force.
2. External dynamics influencing the motion
3. Terminal constraints
4. No inherent performance index in APF guidance

This chapter first seeks to perform a direct comparison of an optimal control solution and an APF generated trajectory on a case study that incorporates all of above items. Once a baseline comparison is complete, the problem will be decomposed to address the effects of saturation, external dynamics, and indirect control individually. Following this, an investigation will be performed to see under what conditions the optimal and APF guidance solutions will exactly match. This will be done by manipulating the terminal constraints of the case study and shaping the artificial potential function. A continuous control APF approach will also be examined. Conclusions will then be drawn about performance, applicability, and practicality of the methods explored.

3.1 Simple Case Study

This case study is one of a sailboat that must cross a fixed width stream as quickly as possible from point A to B (see figure 3.1). The stream has a span-wise parabolic velocity profile (Eq. 3.1) flowing left to right with several sandbars.

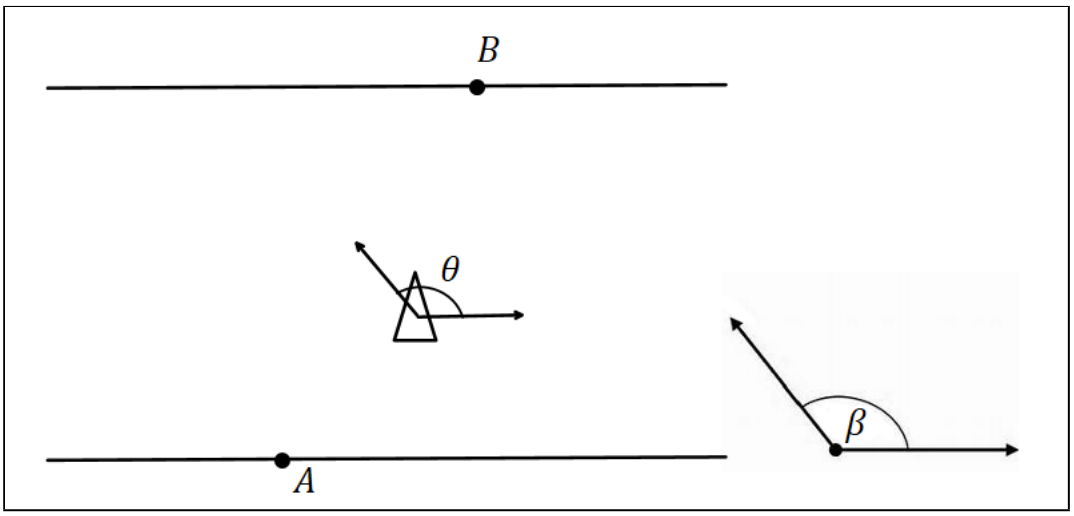

Figure 3.1: Sailboat problem geometry

$$U_{stream}(y) = 4U_{max}\left(\frac{y}{h^2}\right)(h-y) \tag{3.1}$$

where:

U_{max} is the peak span-wise velocity

h is the stream width

y is the y-position of the sailboat

As water flows over the sandbars, constricting the depth, the water is modeled to accelerate due to Venturi-like effects. The sandbars are assumed to be deep enough that boundary layer effects can be neglected and that there is no risk of collision to the sailboat. The velocity increase caused by the sandbars is modeled as a two-dimensional Gaussian function shown in Eq. (3.2). The combined velocity profile due to the sandbars is then simply the sum of each (Eq. 3.3).

$${}^{i}U_{sbar}(x,y) = {}^{i}U_{sbar,max}\exp\left(-\left(\frac{(x-{}^{i}x_c)^2}{2\sigma_{x,i}^2}+\frac{(y-{}^{i}y_c)^2}{2\sigma_{y,i}^2}\right)\right) \tag{3.2}$$

where:

$^{i}U_{sbar,max}$ is the peak span-wise sandbar velocity of the i^{th} sandbar

x is the x-position of the sailboat

$^{i}x_c$ is the x-position of the center of the i^{th} sandbar

y is the y-position of the sailboat

$^{i}y_c$ is the y-position of the center of the i^{th} sandbar

$\sigma_{x,i}$ is the spread of the i^{th} sandbar velocity in the x-direction

$\sigma_{y,i}$ is the spread of the i^{th} sandbar velocity in the y-direction

$$U_{sbar}(x,y) = \sum {}^{i}U_{sbar}(x,y) \tag{3.3}$$

$$U_{tot}(x,y) = U_{stream}(y) + U_{sbar}(x,y) \tag{3.4}$$

The sum of the stream and sandbar velocity profiles represents the total velocity distribution of the stream which is given by Eq. (3.4). The contours of a representative velocity profile are shown in figure 3.2 where the contours represent lines of constant velocity.

The wind speed, W, and direction were fixed with the positive wind direction, β, defined counter-clockwise from the positive x-axis. The problem objective was to minimize the time to cross the stream with the sail direction as the only control. Positive sail angle, θ, was defined as counter-clockwise from the positive x-axis. The sail was assumed to be mounted to a point mass.

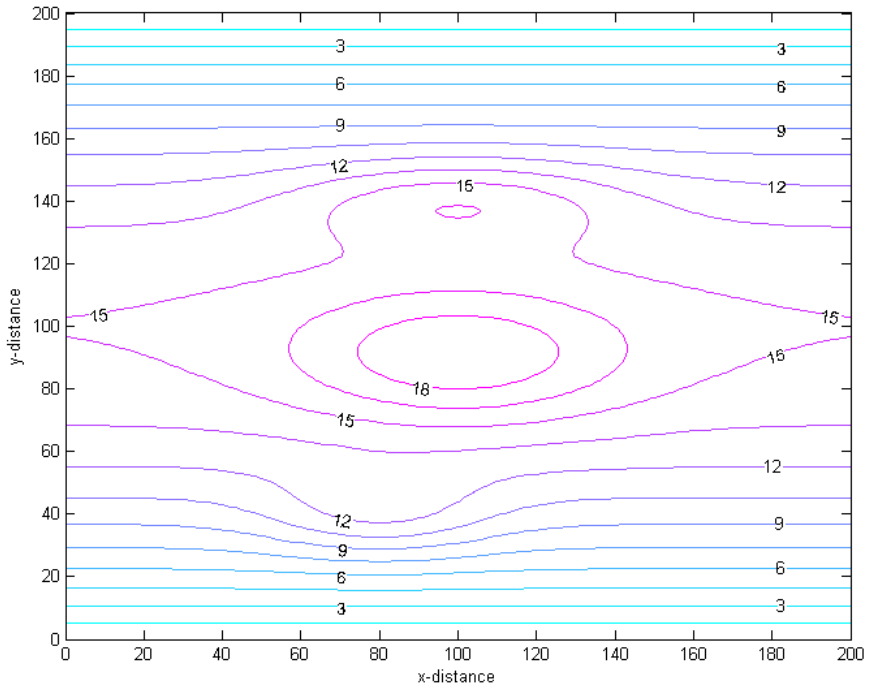

Figure 3.2: Velocity contours

If the wind vector, $\vec{W}$, and sail direction unit vector, $\vec{\Psi}$, are defined as:

$$\vec{W} = W\cos\beta\ \hat{i} + W\sin\beta\ \hat{j} \tag{3.5}$$

$$\vec{\Psi} = \quad \cos\theta\ \hat{i} + \quad \sin\theta\ \hat{j} \tag{3.6}$$

The magnitude of the sailboat's velocity due to the wind is the component of the wind that lies along the sail direction. The total velocity is then the sum of the sailboat's wind velocity and the stream velocity. As evident in Eq. (3.7), the stream velocity only acts in the x-direction and therefore does not contribute to the sailboat's velocity in the y-direction.

$$\dot{\vec{x}} = (U_{tot} + W\cos(\theta - \beta)\cos\theta)\,\hat{i} + W\cos(\theta - \beta)\sin\theta\ \hat{j} \tag{3.7}$$

3.1.1 Optimal Control Solution. The optimal solution was computed as a basis of comparison with the intent of being the "best" possible solution in terms of a performance index. Obviously, the term "best" is subjective, but in this case the assumptions are that the vehicle is operating in a deterministic environment with full knowledge of the system dynamics and that the discretized dynamics are a sufficient representation of the continuous system. Given these assumptions, the optimal control problem for minimum transit time is to minimize the performance index represented by Eq. (3.8).

$$J = \int_0^{t_f} dt = t_f \tag{3.8}$$

Subject to:

$$\dot{\vec{x}} = \left(U_{tot} + W\cos\left(\theta - \beta\right)\cos\,\theta\right)\hat{i} + W\cos\left(\theta - \beta\right)\sin\,\theta\,\hat{j},$$

$$\begin{aligned} x\left(0\right) &= A_x, \quad & x\left(t_f\right) &= B_x \\ y\left(0\right) &= A_y, \quad & y\left(t_f\right) &= B_y \\ \dot{x}\left(0\right) &= 0, \quad & \dot{x}\left(t_f\right) &= 0 \\ \dot{y}\left(0\right) &= 0, \quad & \dot{y}\left(t_f\right) &= 0 \end{aligned}$$

The discretized solution was found using the *fmincon* SQP solver within MATLAB® at the conditions outlined in Table 3.1. The derivation of the discretized equations of motion can be found in Appendix A.

The resulting trajectory is displayed in figure 3.3. The results were intuitive in that the boat uses the wind to sail momentarily upstream in order to avoid being overpowered by the current as velocity increases both midstream and over sandbars. Additionally it was found that for certain geometries (sandbar locations, peak stream velocities, wind speed/direction, and start/end locations) no solution exists. This is

Table 3.1: Problem geometry

	Initial	Final	Sandbar 1	Sandbar 2	Sandbar 3
x	80	120	80	100	100
y	0	200	40	90	140
σ_x	-	-	20	30	30
σ_y	-	-	10	15	10
$U_{sbar,max}$	-	-	2	2.5	1
W_{max}	10	-	-	-	-
β	130	-	-	-	-
U_{max}	10	-	-	-	-

also intuitive because if there is not enough wind available in the appropriate direction, the boat cannot overcome the current. However, if no final x-position is specified, as long as some positive vertical component of the wind exists, a feasible solution always exists.

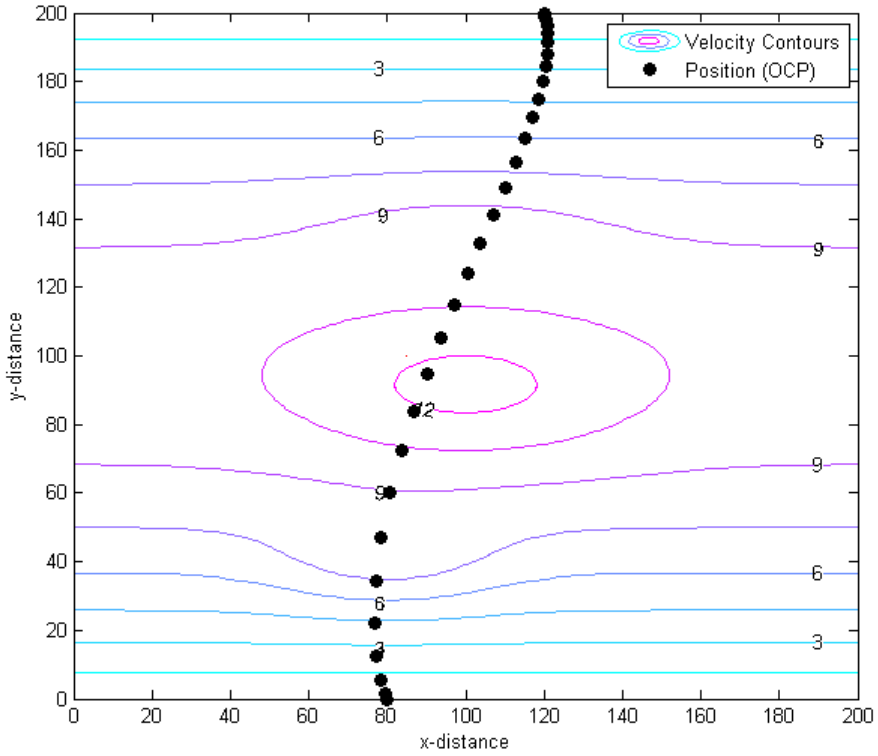

Figure 3.3: Minimum time trajectory

The optimal control solution yielded a transit time of 16.95 seconds. With an inaccurate initial guess, the algorithm took 9.9 seconds of machine time to converge to a feasible solution on an Intel® Core™ i5 1.80 GHz processor. Running the same algorithm initialized with the optimal solution took 0.66 seconds. In either scenario, if the optimal control solution were required to be recomputed at a 10 Hz refresh rate[1], a solution would not be available by the time a guidance solution was needed. In attempt to improve computational efficiency, a traditional APF guidance solution was implemented next and the performance index was compared.

3.1.2 APF Solution. A simple quadratic potential field was implemented in attempt to guide the sailboat to its desired end state. Equation (3.9) contains the mathematical representation for the APF.

$$\phi = \frac{1}{2}\vec{r}^{T} Q \vec{r} \tag{3.9}$$

where:

Q is a positive semidefinite weighting matrix

$\vec{r} = [x - x_f, y - y_f]^T$

A typical APF control law is represented by figure 3.4. This control law is accurate for systems in which a change in velocity ($\Delta\vec{v}$) can be directly applied. For systems like the sailboat, there is an extra step to determine what control will result in the required $\Delta\vec{v}$.

Obtaining a viable APF solution was a challenge due to the external forces influencing the trajectory. Much research has been accomplished for systems without external dynamics [3, 7]. However, with the sailboat problem, the inclusion of an external force (the stream current) meant that direct determination of the control

[1]While a 10 Hz refresh rate would not be necessary given the actual velocity of the sailboat, it would not be an unrealistic requirement for systems with faster dynamics.

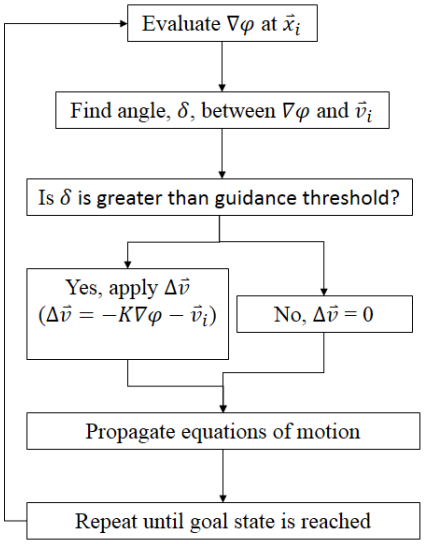

Figure 3.4: Artificial potential function control law diagram

(sail angle) was not possible. This is because the force required by the APF is driven by a change in velocity, but the control is sail angle. To solve for the control, it was assumed that the $\Delta\vec{v}$ command was achievable, although this is not always true. With this assumption, the position at the next step was then determined. The position was required in order to calculate the velocity of the stream at that point since the distribution was non-uniform. Unfortunately, since the $\Delta\vec{v}$ is not necessarily achievable, the sailboat may not actually reach that position. Given the assumed $\Delta\vec{v}$, the new position, and the stream velocity at that point, the required sail angle was then computed. If no solution existed, the sail angle that caused the velocity vector to be the closest to the commanded $\Delta\vec{v}$ was used instead.

A second issue arose because this method required $\Delta\vec{v}$ to have both direction and magnitude in order to determine sail angle. This posed a problem because with the current APF implementation, $\Delta\vec{v}$ magnitude is a function of the APF gain which was empirically set. Therefore a method must be created of reliably computing the $\Delta\vec{v}$ magnitude such that the external forces may be reliably overcome.

Various "tuning" techniques were employed in order to match the APF solution to the OCP solution as closely as possible. For the case with no obstacles, the APF parameters that can be manipulated are the gain (K), weighting matrix (Q), and time step (Δt). Ideally, sensitivity to the time step should be minimized. Initial experimentation included adjusting refresh rate and gain. The APF trajectory shown in figure 3.5 is the result of tuning a time invariant gain.

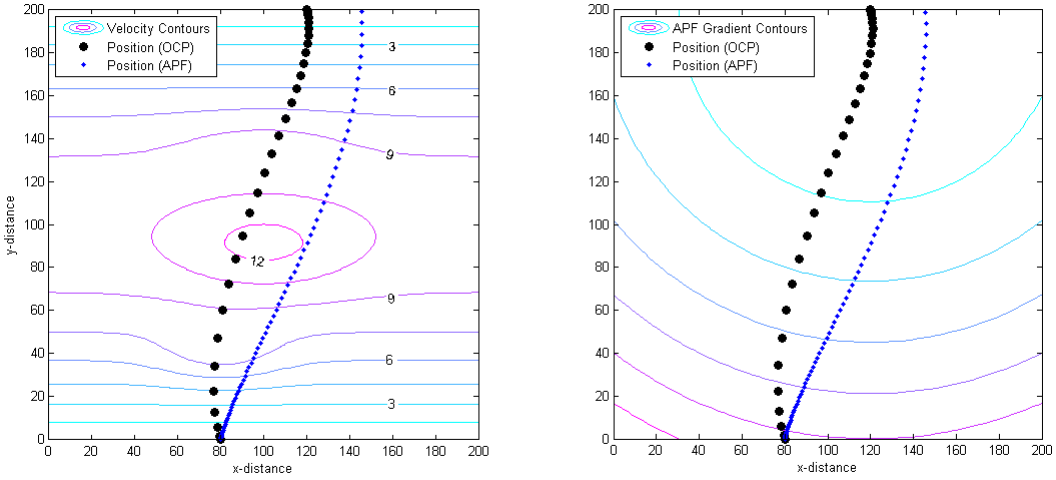

Figure 3.5: Artificial potential function and optimal control solutions

Ultimately, the APF algorithm took 3.42 seconds to complete on the same computer. However, in a real-time guidance sense, the entire trajectory is not needed before an APF generated control can be implemented since the control at each time step is a strict function of the local potential field gradient. This is contrary to the OCP solution which by its very nature requires that the entire trajectory satisfy all constraints before the entire control sequence is generated. Therefore, in the spirit of real-time guidance, a better comparison would be the time required to compute each step which on average took 0.0124 seconds. While the APF algorithm has a clear advantage in computational efficiency, the optimal control solution has the advantage with respect to the performance index. Although the APF does not attempt to minimize the chosen performance index, the end result can still be compared to that of the OCP. The APF solution resulted in a total transit time of 14.4 seconds. While

this is faster than the OCP solution, it is important to note that the APF violated the terminal constraints outlined for the OCP in both position and velocity. This is due to the absence of feedback and control saturation in the APF algorithm.

The above comparison does not in fact represent the most accurate comparison between the two guidance methods. The OCP solution was meant to represent the best possible solution with respect to the cost function given that the stream velocity profile is known. With such a solution known, it would be desirable to determine how the APF guidance algorithm may be manipulated to duplicate the results.

To more accurately compare the algorithms from a real-time guidance perspective, the OCP solution can be recomputed without the stream velocity profile known. This is based on the assumption that in a realistic scenario, a controller would not have explicit knowledge of the external dynamics present. The control sequence that minimized transit time would then be used to propagate the full equations of motion. In this scenario, the new OCP solution is likely to lead to performance worse than that of the APF since the knowledge of the intended goal state is lost[2]. This scenario is likely if the velocity profile is unknown, a varying wind is present, or stochastic variations or uncertainties exist within the system dynamics. However, the APF retains knowledge of the goal via the minimum in the potential field and directs the vehicle toward the minimum at every iteration.

The propagated OCP solution crossed the stream in 13.29 seconds and took 5.93 seconds to compute. As predicted, figure 3.6 shows that the OCP solution performs worse than the APF solution when it has no knowledge of the stream dynamics.

The left side of figure 3.6 shows the APF-generated trajectory and the trajectory from the propagated OCP control sequence overlaid on the stream's velocity contours. The right side represents the same trajectories overlaid on the contours of the artificial

[2]It should be noted that there are ways to formulate the optimal control solution to account for such uncertainty without propagating an "ignorant" OCP-generated control sequence, but it was the author's intent to demonstrate how APF performance falls between a best case and worse case optimal control problem.

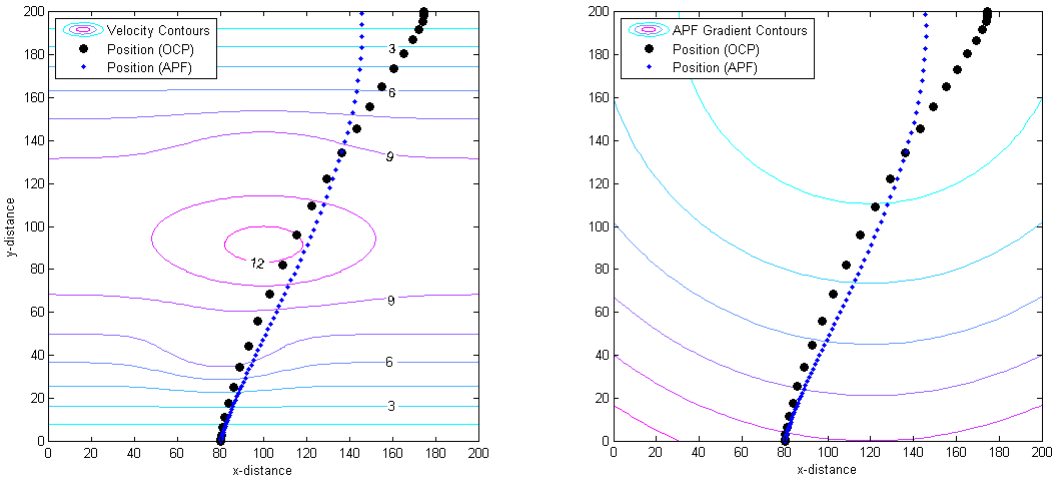

Figure 3.6: Artificial potential function and optimal control solutions with no knowledge of stream current

potential field. Table 3.2 summarizes the comparison between all three proposed trajectories.

Table 3.2: Results summary for APF and OCP guidance

	Final x-position	Final y-position	t_f	Total Run Time
OCP	120.0	200	16.95	9.90
Propagated OCP	174.7	200	13.29	5.93
APF	145.7	200	14.40	3.42

3.2 Parametric Study

Before attempting a hybrid method in support of the primary research objective, it is important to further characterize APF performance. A parametric study was performed on various APF formulations by varying the peak span-wise velocity of the stream's parabolic profile. The velocity variation due to sandbars was omitted in all cases leaving solely a parabolic velocity variation. The first part of the study set equal APF gains, shaping parameters, and refresh rates. For the initial phases, the sailboat was exchanged for a motorboat with thrusters and unlimited control authority was assumed. The intent of this phase was to observe the effect of increasing external forces on the gradient-following capability. The second phase implemented

a maximum allowable thrust to examine the effects of both external forces and saturation. Additionally, multiple types of attractive potentials were examined in each phase.

3.2.1 Motorboat Phase 1. Phase one consisted of varying peak span-wise velocities with no restriction on control authority by allowing unlimited thrust to observe performance of both Gaussian and quadratic APFs. Trajectories were generated from a Gaussian attractive potential given by Eq. 3.10.

$$\phi_a(x,y) = -\lambda \exp\left[-\left(\frac{x-x_o}{\alpha} + \frac{y-y_o}{\beta}\right)\right] \tag{3.10}$$

Table 3.3 lists the conditions under which both algorithms were run. There was assumed to be no influence from the sandbars in the parametric study.

Table 3.3: Summary of parametric study test conditions

W_{max}	10
β	130
$U_{sbar,max}$	none

Figure 3.7 (A) demonstrates that when the stream velocity is zero, the motorboat can perfectly descend along the APF gradient and reach the goal with zero velocity. Figures 3.7 (B-D) show that as velocity is increased and all other parameters remain unchanged, the motorboat can no longer follow the negative gradient of the APF. This is because the system dynamics are unaccounted for in the APF formulation. When the force is applied and the equations of motion are propagated forward, the stream current forces the motorboat downstream of its commanded position. This can be observed in figure 3.8 which is a zoomed in version of figure 3.7 (B).

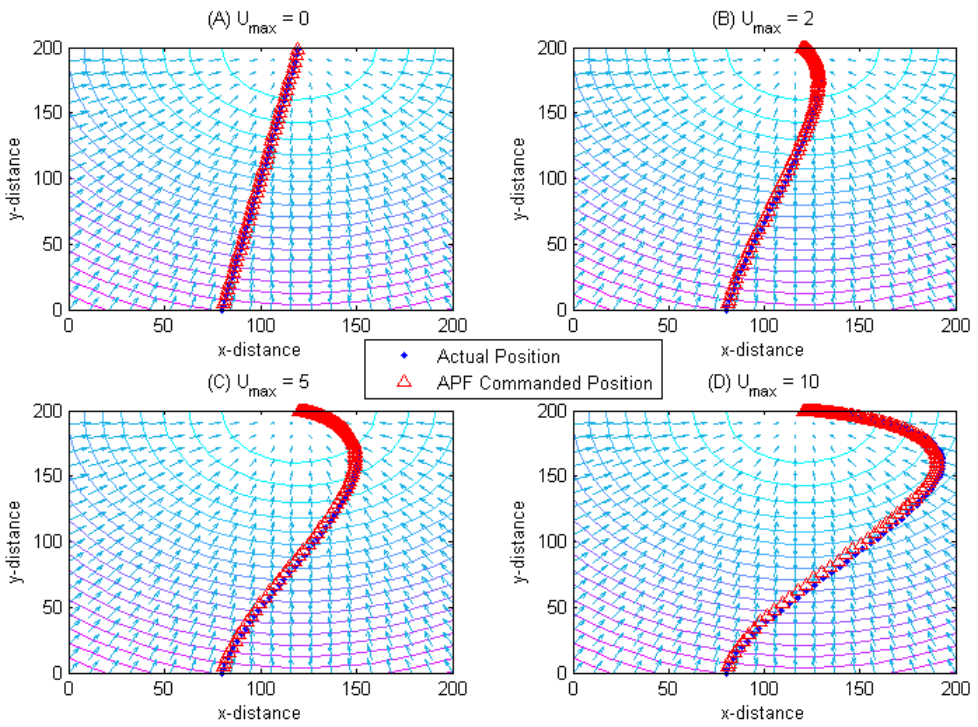

Figure 3.7: Gaussian attractive potential for varying peak velocities and unsaturated control

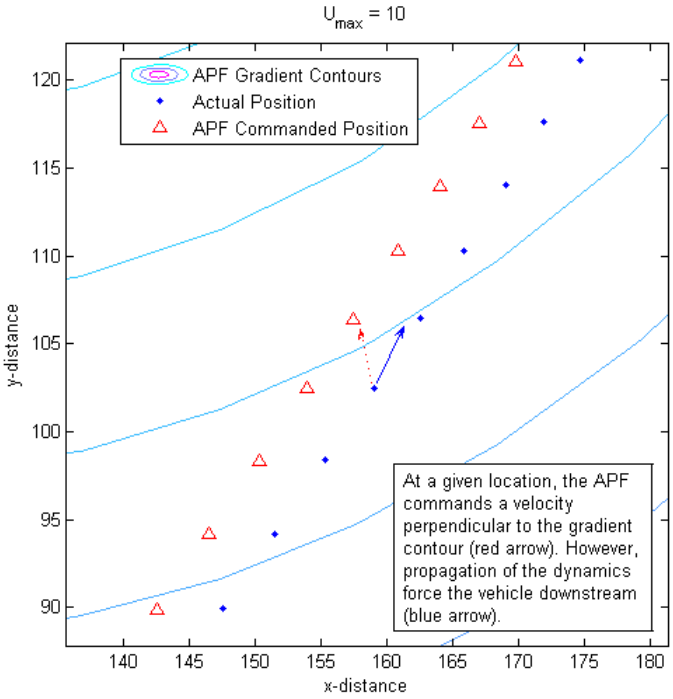

Figure 3.8: Gaussian attractive potential, $U_{max} = 10$, commanded vs. actual position

In figure 3.8, the red triangles represent the boat position if the APF-commanded $\Delta\vec{v}$ for that particular time step is propagated with no external forces. By inspection, the associated velocities are aligned with the negative gradient of the APF. The blue dots represent the actual boat position when applying the commanded $\Delta\vec{v}$ and propagating the full equations of motion. Ideally, these markers would lie on top of one another.

It is interesting to note that as the velocity of the stream approaches zero, the APF-generated forces dominate and guide the motorboat back to the goal. Unfortunately, this behavior cannot be generalized for all velocity profiles. For example, a uniform stream velocity distribution would not decrease toward zero near the goal and the APF generated forces alone would be be insufficient to guide the motorboat to the goal.

If the APF gain is increased high enough and the time step is decreased small enough, the trajectory approaches that of figure 3.7 (A) (i.e. perfect gradient descent). The problem in doing so is that the gains and time steps required to compensate for the system dynamics are physically unrealizable. However, even if perfect gradient descent is not practical or possible, benefits of employing APF guidance may still exist. Namely, if the APF-generated force is sufficient relative to the system dynamics, convergence and obstacle avoidance may still be possible. Figure 3.9 shows that similar results were obtained when a quadratic attractive potential (Eq. 3.9) was used.

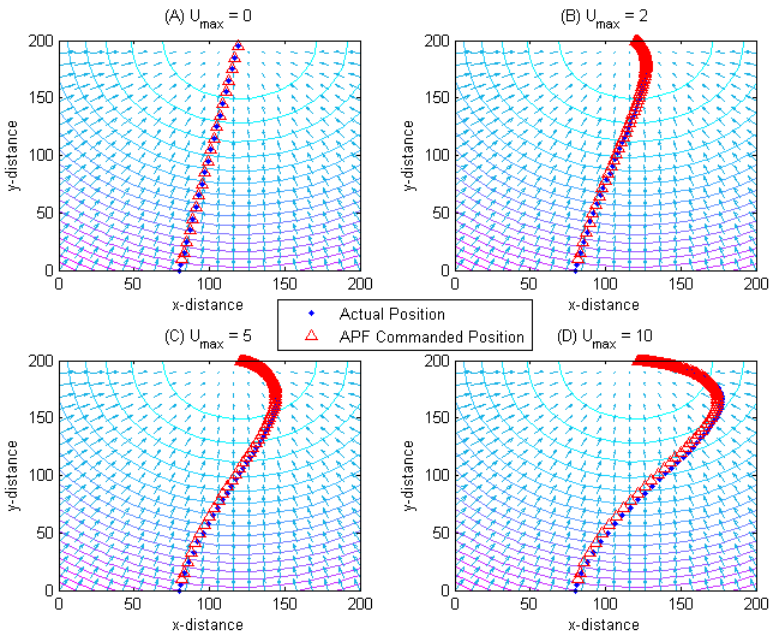

Figure 3.9: Quadratic attractive potential for varying peak velocities and unsaturated control

3.2.2 Motorboat Phase 2. This phase mirrored the previous analysis, but a maximum allowable thrust was introduced to examine the effects of control saturation. In each case, the algorithm limited the maximum $\delta\vec{v}$ per step to 0.1. Since the vehicle was assumed to be a point mass, this was equivalent to a maximum allowable thrust.

By inspection of figure 3.10 (A), it would appear that perfect gradient descent is possible even with the saturated case for a stream velocity of zero. However, examining the motorboat velocity profile shows this not to be the case. While the trajectory projection onto the 2D plane is in fact aligned with the negative gradient of the APF, the thrust saturation prevents the magnitude of the desired velocity from being captured. This causes a violation in the terminal constraint on velocity. This is more obvious by examining figure 3.10 (B-D) which are unable to capture the desired goal in both position and velocity. Additionally, if the gradient cannot be followed, then no guarantees exist for obstacle avoidance. As with phase 1, the quadratic attractive potential displayed similar results (Fig. 3.11).

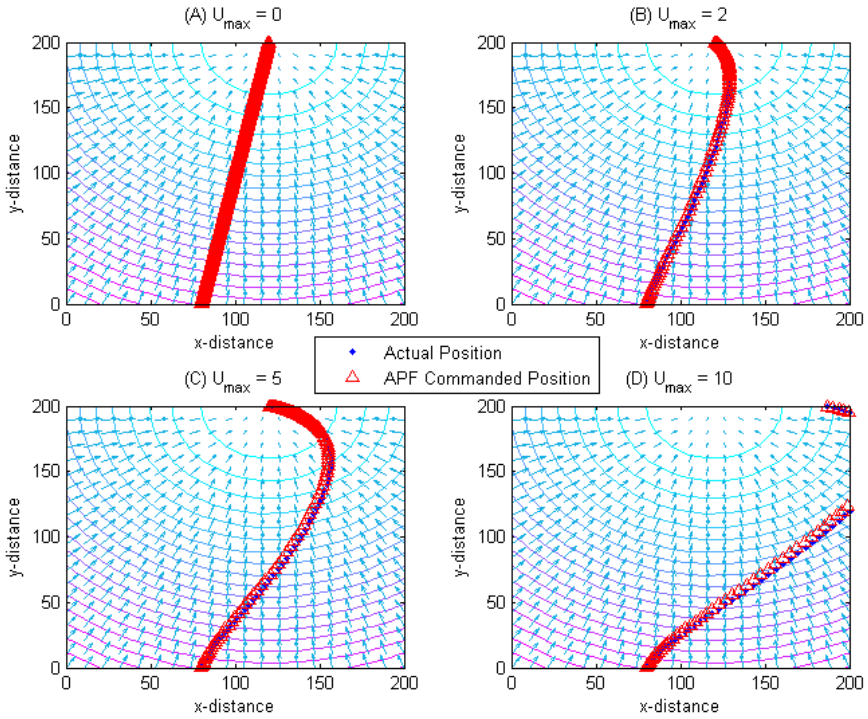

Figure 3.10: Gaussian attractive potential for varying peak velocities and saturated control

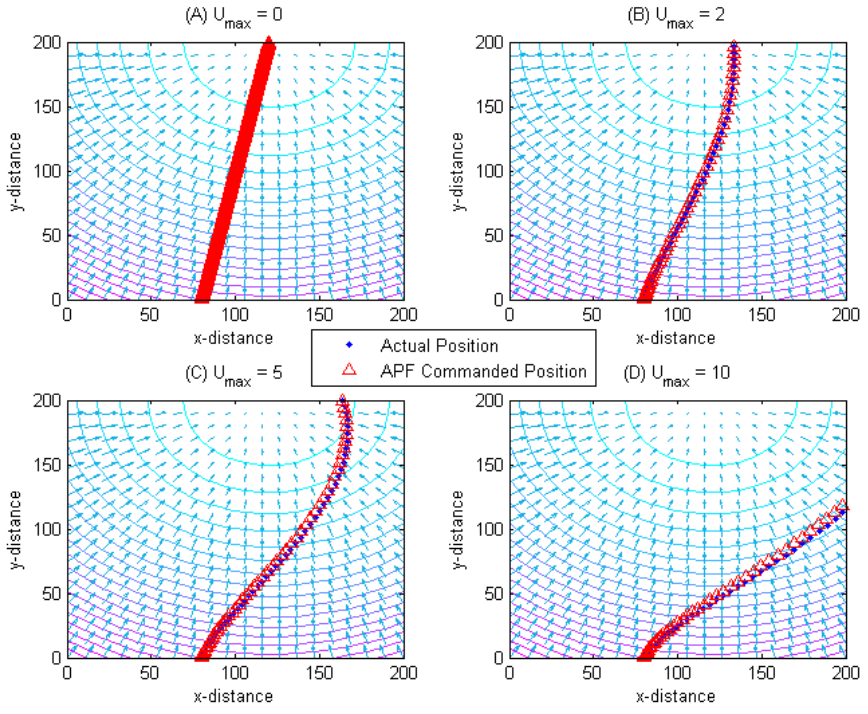

Figure 3.11: Quadratic attractive potential for varying peak velocities and saturated control

Obviously, control saturation combined with external forces can negatively effect convergence of the guidance algorithm to the desired state. It is possible this negative impact on convergence can be avoided if the algorithm is designed to perform a reachability analysis and use the results to adjust the potential function such that the desired states are achievable.

Table 3.4 summarizes the results of each of the studies. The unrealistic transit times of the unsaturated $U_{max} = 0$ cases should be noted. The $\Delta\vec{v}$ commands required by the algorithm to exactly follow the negative gradient of the potential field are quite large and in may cases would be physically unrealizable. Similarly, the $U_{max} = 0$ saturated cases have a much slower transit time. This is because of the small $\Delta\vec{v}_{max}$ chosen for illustration purposes. In the remaining cases, the stream current helps to accelerate the motorboat and faster transit times are observed.

Table 3.4: Summary of parametric study results

Phase 1 (Unsaturated)				
Case	APF Type	Transit Time (sec)	Total Distance	X-distance from Goal
$U_{max} = 0$	Gaussian	4.7	203.96	0
	Quadratic	2.1	203.96	0
$U_{max} = 2$	Gaussian	20.0	209.72	0
	Quadratic	14.5	208.38	0
$U_{max} = 5$	Gaussian	20.1	231.86	0
	Quadratic	14.5	225.48	0
$U_{max} = 10$	Gaussian	20.5	289.08	0
	Quadratic	14.5	265.78	0
Phase 2 (Saturated)				
Case	APF Type	Transit Time (sec)	Total Distance	X-distance from Goal
$U_{max} = 0$	Gaussian	204.3	203.96	0
	Quadratic	204.2	203.96	0
$U_{max} = 2$	Gaussian	55.3	210.50	0.01
	Quadratic	22.6	209.86	13.42
$U_{max} = 5$	Gaussian	55.1	239.04	0.03
	Quadratic	22.1	223.21	33.47
$U_{max} = 10$	Gaussian	55.1	276.33	185.60
	Quadratic	21.9	256.59	106.84

3.2.3 Sailboat. For vehicles such as the sailboat, an indirect control must be introduced within the APF formulation. Continuing the parametric study to the sailboat allows the effects of indirect control to be observed. It is expected based on the initial results in Section 3.1 that the results will be similar to the saturated motorboat study. Figure 3.12 shows how the sailboat trajectory varies as a function of stream velocity.

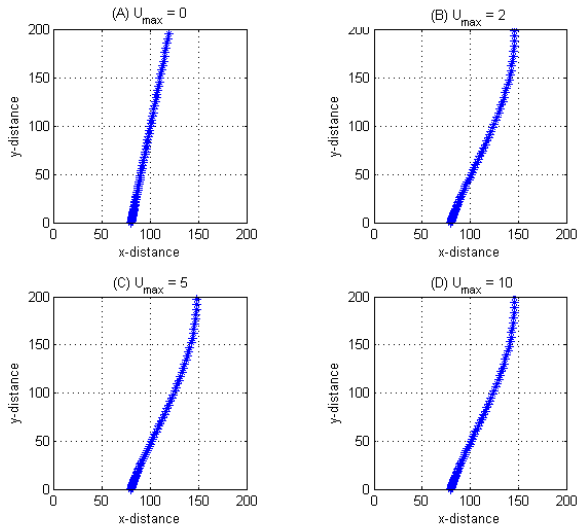

Figure 3.12: Sailboat trajectory for quadratic APF guidance

Using sail angle as a control essentially saturates the maximum allowable $\Delta\vec{v}$ where the maximum is a function of the wind speed, W, and sail angle minus wind direction, $\theta - \beta$. It is reasonable to assume that with a high enough wind, the sail-controlled results would approach that of the unsaturated motorboat parametric study. However, without an additional control for drag, the sailboat has no means to slow down while also descending along the APF gradient. It is also important to note that due to the $\theta - \beta$ functionality of the saturation, the maximum allowable $\Delta\vec{v}$ is variable. Therefore, the wind speed would need to be much higher (and time step much smaller) than its parametric study counterparts and would greatly depend on wind direction. While these are noticeable differences between the studies, the

conclusion is the same: convergence and obstacle avoidance cannot be guaranteed without further modification to the algorithm.

In addition to saturation, the computational burden of the solution increases because the algorithm must solve a system of nonlinear equations at every time step to determine the required angle. Experimentally, this has not shown to be a significant increase in computation time. This is likely because the equations often have no solution due to the saturation induced by indirect control power. When no solution is found, the algorithm points the sail along the unit vector in the direction of the new desired velocity even though the magnitude is unattainable. If drag is included as an additional control, then the issue of how to distribute control power is introduced. This problem is not unique to APF guidance, but nonetheless must be adequately addressed for the algorithm to successfully converge upon the goal.

3.3 Trajectory Matching via APF Tuning

As evident in Section 3.1, tuning the APF parameters, i.e. adjusting the APF gain and strength, did not provide an exact match to the OCP solution. This explores whether eliminating some of the constraints of the sailboat problem will result in better matching of the two trajectories. The influence of the sandbars was removed and the maximum stream velocity was gradually decreased to zero. As the span-wise velocity approached zero, the lack of external forces influencing the dynamics allowed the APF to guide the sailboat exactly along the negative gradient of the potential field and reach the goal state as demonstrated in Section 3.2. However, the result was still suboptimal when the transit time was compared with the optimal control solution because the optimal path uses favorable winds to improve the performance index.

In attempt to further simplify the problem, the constraint on final x-position was removed. With this scenario, the final position from the optimal control solution was upstream from the starting position since it caused the sailboat to sail into the wind to increase velocity. The APF, however, traversed a line perpendicular to the shore

with the final x-position equal to the starting x-position. Both results are pictured in figure 3.13. With no x-position specified for the goal, the negative gradient of the potential function points perpendicularly toward the $y = h$ contour at every point. In general, this behavior increases the transit time of the sailboat under APF guidance, but will only exactly match the optimal trajectory when $\beta = 90°$.

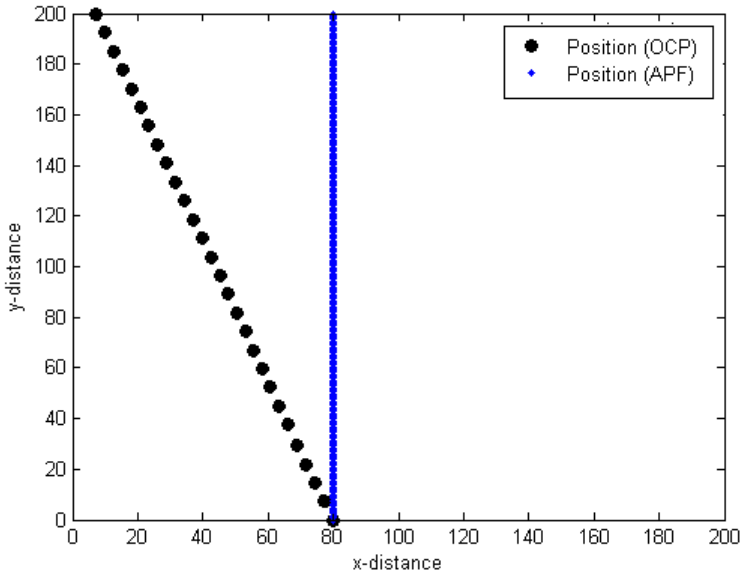

Figure 3.13: Sailboat path with APF and OCP free final x-position

Noting that the APF can be optimal under this condition, the APF can be slightly reformulated to force the gradient parallel to the wind direction. While leaving the OCP solution as open final x-position, the APF goal state can be specified as the final x and y position of the OCP solution (see figure 3.14). Clearly this behavior, while desirable, could only be achieved with gross simplifications to the original problem statement.

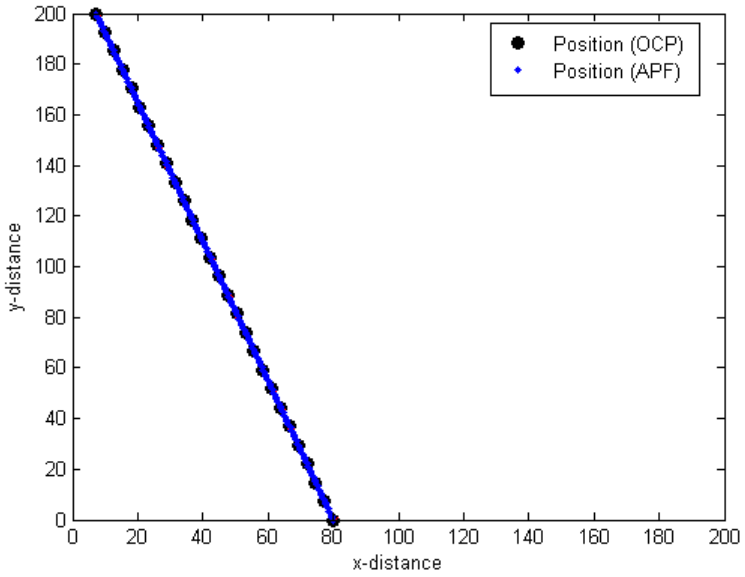

Figure 3.14: Sailboat path with APF fixed final x-position and OCP free final x-position

3.4 Potential Function Fitting

From Section 3.3, it is clear that for time invariant APF gains, the trajectory can only match the optimal solution in very limited applications with gross simplifications to the problem statement. However, it may be possible to fit an APF trajectory to a desired optimal trajectory.

3.4.1 Static APF Parameters. The APF trajectory is influenced by the location and strength of repulsive and attractive potentials, therefore by adding pseudo-obstacles and/or intermediate goal states it may be possible to more closely match a known trajectory with static shaping parameters.

The number of pseudo-obstacles being introduced was varied with the assumption that more obstacles would allow for closer matches. The radius and position of the obstacles were optimized by minimizing the difference between the APF and OCP trajectories. At first glance, this methods appears to trade one optimal control problem for another. While this is true, there are possible advantages to be had in doing so. By optimizing the parameters of the pseudo-obstacles, it is conceivable that

the formulation of the optimal control problem may be simplified thus providing some increase in computational efficiency.

The repulsive potential of each obstacle was held constant, but may be left free to vary if so desired. First the pseudo-obstacles were distributed evenly in the y-direction and the x-position was optimized. For a fixed y-position, the repulsive potential for each obstacle has $2n$ parameters to optimize, where n represents the number of obstacles. For a free y-position, this will double to $4n$. However, in either case the time to compute the optimal positions should be greatly decreased from the time to compute the optimal control trajectory. This is partly because the number of parameters to optimize is greatly decreased. More importantly, the computation time should decrease because it is now a static optimization problem. The APF trajectory was constrained to finish at the x and y position of the goal.

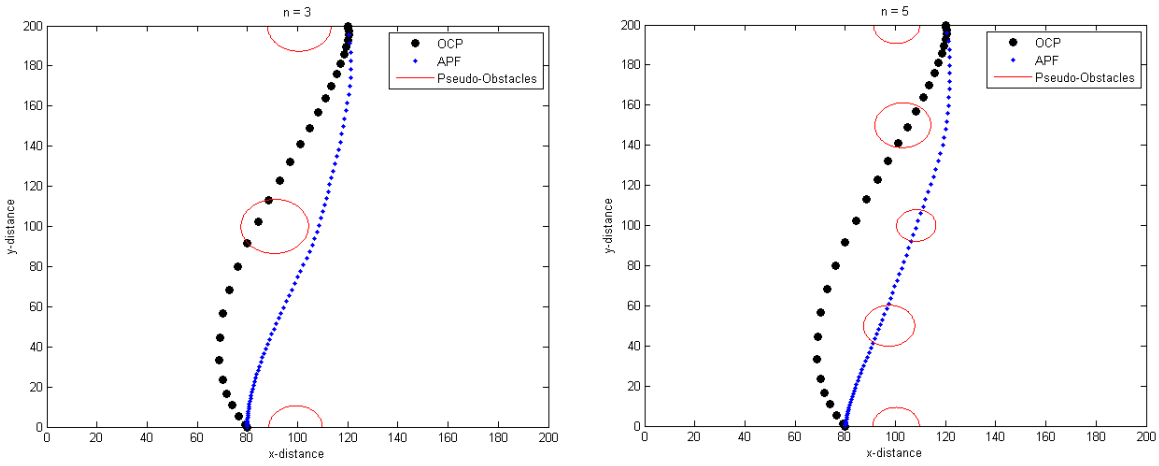

Figure 3.15: Optimized pseudo-obstacle position for fixed y location

While better APF-generated trajectories were obtained as indicated in figure 3.15, adding pseudo-obstacles proved to be an ineffective way of trajectory matching due to extended computation time. All run times were upwards of 300 seconds. This was contrary to expectations due to the small number of parameters to optimize. Additionally, the matching did not reliably contribute to the convergence of the APF to the final goal in terms of velocity without substantially burdening the optimizer. This was because to ensure convergence, the optimizer required final velocity constraints in

addition to final position constraints. When the additional velocity constraints were implemented, the optimizer was not able to find a feasible solution.

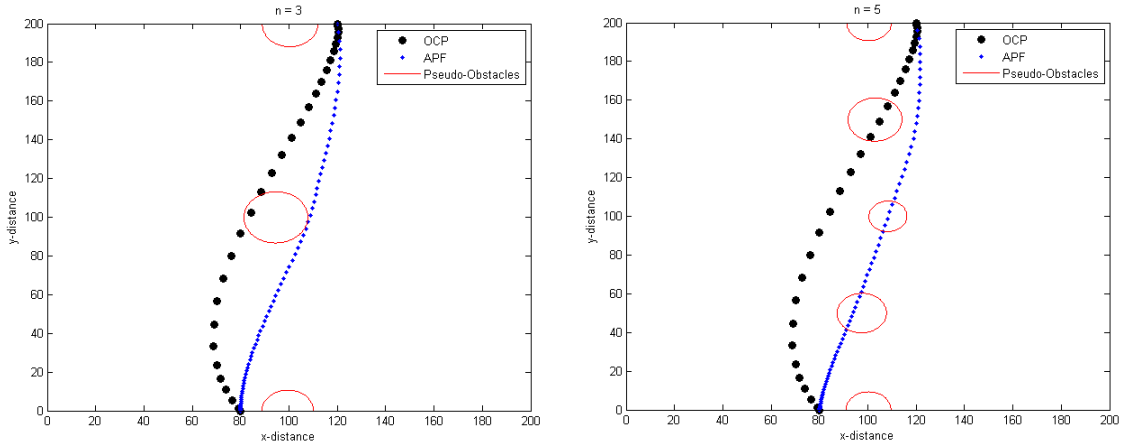

Figure 3.16: Optimized pseudo-obstacle position for free y location

As predicted, the matches improved with the number of obstacles added but not as much as expected. In general, the more obstacles added, the closer the fit can be made. However, this increases the number of parameters to be optimized thus increasing computation time substantially. Surprisingly, the results did not vary substantially when the y position of the obstacles was left free to vary. Figure 3.16 shows the solutions for both $n = 3$ and $n = 5$ for free y position. While it is quite possible there is simply a greater dependence on x position and radius, this may also be indicative of the optimizer finding local minima instead of global minima in the solution space. The latter seems more likely given that the obstacles did not vary greatly from their initial positions. This was especially true for the y positions of the obstacles. Table 3.5 summarizes the output from the obstacle position optimization.

Table 3.5: Pseudo-obstacle optimization summary

	Fixed y			Free y		
$n = 3$	r	x	y	r	x	y
	10.64	99.18	0	10.69	99.40	-0.03
	13.48	90.87	100	13.22	94.55	99.93
	12.69	100.716	200	11.48	100.46	199.57
	Fixed y			Free y		
$n = 5$	r	x	y	r	x	y
	9.33	100.22	0	9.80	100.12	-0.03
	10.22	97.50	50	9.70	96.54	49.50
	7.89	108.30	100	7.70	108.60	103.24
	11.19	102.95	150	11.36	103.78	149.90
	9.27	100.50	200	8.96	100.46	200.78

Trajectory matching in this fashion also requires significant off-line computation time to find the optimal trajectory which would likely be detrimental to real-time guidance applications. Alternative methods that can correct the trajectory en-route would be more desirable in terms of computation time.

3.5 Continuous Control

While gradient descent is highly desirable due to its computational efficiency, the methodology has no inherent knowledge of the system dynamics other than the satisfaction of dynamics from the forward propagation of states. One approach to remedy this deficiency is to develop a continuous control solution. A desirable continuous control solution uses knowledge of the system dynamics along with gradient descent to drive the vehicle to its desired end state. Such a method was developed by Fields who defined a velocity potential such that $\dot{\vec{r}}$ is driven to $-\nabla\phi$ [15].

$$\phi_v = \frac{1}{2}\left(-\nabla\phi - \dot{\vec{r}}\right)^T \left(-\nabla\phi - \dot{\vec{r}}\right) \tag{3.11}$$

where:

ϕ_v is the velocity potential

ϕ is the position potential

$\vec{r}$ is the position vector

Equation (3.11) is a quadratic defined in the velocity space with a minimum at $-\nabla\phi$. By taking the time derivative of the velocity potential, Eq. (3.12) is obtained.

$$\dot{\phi}_v = \left(-\nabla\phi - \dot{\vec{r}}\right)^T \left(-H\left(\phi\right)\dot{\vec{r}} - \ddot{\vec{r}}\right) \tag{3.12}$$

where $H\left(\phi\right)$ is the Hessian of the position potential

Fields then defined the acceleration as:

$$\ddot{\vec{r}} = -H\left(\phi\right)\dot{\vec{r}} - K_d\left(\nabla\phi + \dot{\vec{r}}\right) \tag{3.13}$$

where K_d is a positive definite diagonal gain matrix

Substituting Eq. (3.13) into Eq. (3.12) then yields the following expression for the time rate of change of the velocity potential:

$$\dot{\phi}_v = -2\left(-\nabla\phi - \dot{\vec{r}}\right)^T K_d\left(-\nabla\phi - \dot{\vec{r}}\right) \tag{3.14}$$

Fields noted that by inspection, the solution of Eq. (3.14) is a decaying exponential when K_d is positive definite, thus the requirement for $\dot{\vec{r}}$ to be driven to $-\nabla\phi$ is satisfied. The continuous control law results from substituting Eq. (3.13) into the system dynamics and solving for the required control.

Like the APF methods previously applied in this research, the continuous control solution is suboptimal. However, since the solution has built-in knowledge of the system dynamics, it will converge exponentially to the goal state. It is important to note that in order to employ the continuous control solution described in Eq. (3.11), the system must be fully actuated, i.e., direct control of the vehicle's acceleration must be available. Therefore, this method is applicable to the motorboat problem, but not the sailboat.

The motorboat problem will be studied to examine whether the continuous control solution will indeed account for the dynamics caused by the stream current and drive the vehicle to the goal.

For the motorboat, the system dynamics are defined as:

$$\vec{v}_i = (U + V_{x,i})\,\hat{i} + V_{y,i}\hat{j} \tag{3.15}$$

$$\vec{x}_{i+1} = \left(\dot{\vec{x}}_i + \Delta\vec{v}_i\right)\Delta t \tag{3.16}$$

where U is the stream velocity and

$V_{x,i}, V_{y,i}$ are the x and y components of the motorboat velocity

The resulting trajectory of the motorboat is shown in figure 3.17. The time histories (as shown in figure 3.18) clearly show exponential convergence to the goal states.

The rate at which the convergence occurs is directly related to the gain matrix K_d and the shaping of the position potential as defined by Q. Higher values of the K_d diagonals cause the goal state to be approached more rapidly, but do so at the expense of control. Similarly, as Q steepens the position potential, the goal will be reached more quickly.

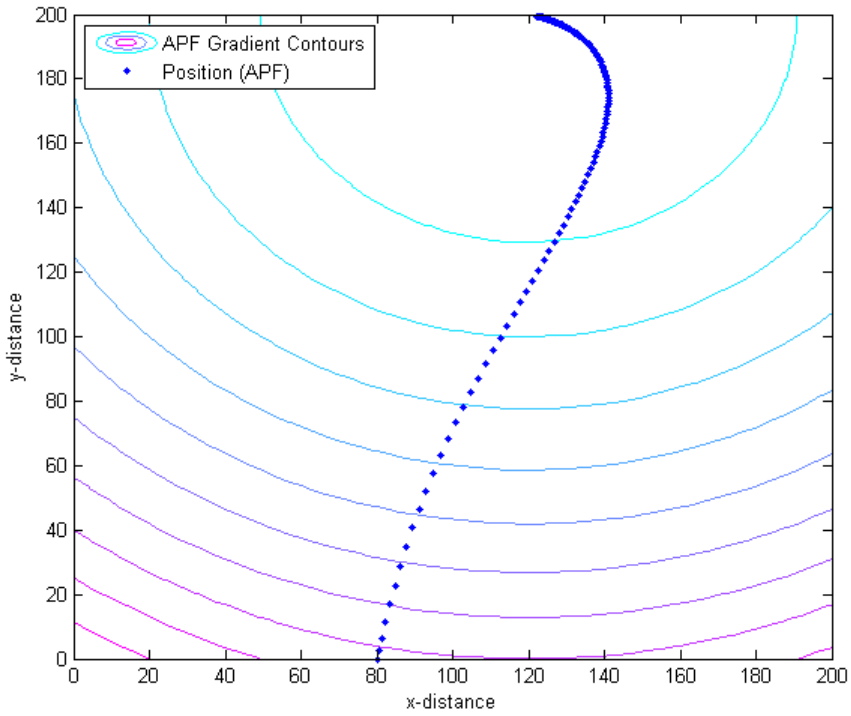

Figure 3.17: Motorboat trajectory using continuous control

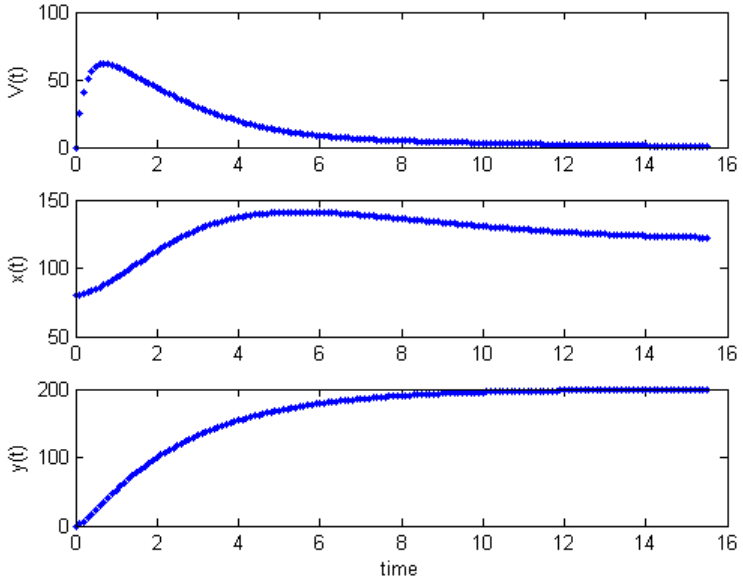

Figure 3.18: Motorboat time histories using continuous control

3.6 Summary

The results have shown that for the chosen velocity profile, multiple forms of APF guidance can guarantee convergence to the goal if the vehicle is fully actuated and unsaturated. However, as saturation increases, the likelihood of convergence decreases. This is not to say that convergence with saturation is impossible, but it is greatly dependent on the amount of saturation present and the external dynamics. It was also shown that the indirect control actuation essentially behaved as a variable saturation. Furthermore, it has been observed that behavior is dependent on the nature of the dynamic environment.

Additionally, alignment of the OCP and APF generated trajectories was shown to be possible, but the circumstances under which it occurs is so limited that it would not be practical for general use. The results also showed that the APF can be statically shaped by introducing false obstacles, but the process is too computationally expensive to be practical.

Embedding the system dynamics via the continuous control method proved to be the only way to guarantee convergence, but its use is limited to fully actuated systems. Each of these findings support both the performance characterization portion of the primary research objective and the secondary objective of determining the suitability of APFs for different classes of problems.

The computational benefit of APF guidance was demonstrated by the quick run times recorded, however, the bulk of the discussion on APFs with respect to cost function will be relegated to Chapter 4. This is because that while total cost is important and of interest, it is irrelevant without convergence. Therefore, in support of the primary goal of this research, Chapter 4 will develop and implement a hybrid algorithm that incorporates both APF guidance and optimal control with the intent of improving both convergence and cost for saturated control in a dynamic environment.

IV. Hybrid Algorithm Development and Application

This chapter details the development of a hybrid guidance algorithm that uses both receding horizon and artificial potential function planning with the intent of addressing the weaknesses of APF path planning outlined in Chapter 3. The utility of the algorithm will be assessed based on its application to the case studies introduced in the previous chapter.

4.1 Receding Horizon

As mentioned in Chapter 2, a receding horizon (RH) solution approaches the optimal control solution as the horizon, h, approaches t_f. In practice, h is much less than t_f. The optimal control must be determined at each $t_0 + h$. The control, which is discretized over each horizon, is then applied over some $\delta < h$. The process then repeats until the desired end state is reached. Figure 4.1 represents the RH process pictorially.

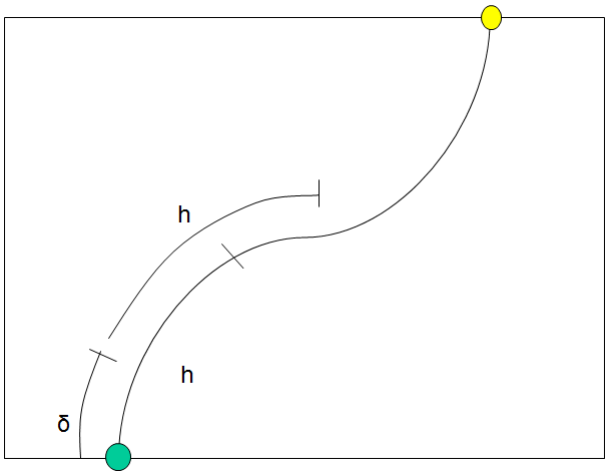

Figure 4.1: Receding horizon planning illustration

The benefit of this method is that it is typically much faster to repeatedly compute optimal control solutions over a subset of the problem than to compute the optimal control solution over the entire trajectory space at once. Additionally, it is better suited to real time guidance since a smaller OCP must be computed at each update. It is also more desirable when external dynamics or uncertainty are relevant.

For example, the sailboat (or motorboat) guidance algorithms have no knowledge of the stream current. As shown in the parametric study in Chapter 3, an APF algorithm with no knowledge of the system dynamics is subject to drift and faces loss of convergence guarantees. It was also observed that for a system with no saturation, convergence of the APF trajectory was guaranteed but performance (in terms of a cost function) was lacking.

While RH planning seems like an excellent alternative to the optimal control method previously presented, there must be an effective method for estimating the cost-to-go (CTG). The cost-to-go refers to the cost to get from $x(t+\delta)$ to $\vec{x}(t_f)$. Henshaw suggested that an appropriate measure of CTG was the value of the attractive potential at $x(t+\delta)$. This formulation guarantees that performance will be no worse than that of the baseline APF. However, as evident from previous results, the baseline APF alone is inadequate for guarantees on performance when external dynamics influence the vehicle's motion. One of the other drawbacks of receding horizon planning is that it shares some of the weaknesses of optimal control path planning. Perhaps the most important weakness is that, in general, there is no guarantee a feasible solution will be found. Because of this major drawback, a supplemental path planner would be necessary to ensure robustness.

Given these observations, the weaknesses of both APF and RH path planning may be mitigated by directly supplementing an RH planner with APF guidance. Assuming a feasible solution is found, the RH planner will then perform at least as well as if not better than the APF planner via the potential field evaluation in the cost function. In the absence of a feasible solution, the APF planner will take over for the next horizon.

4.2 Hybrid Guidance

A hybrid algorithm was developed first for the unsaturated, fully actuated motorboat problem to perform the following actions:

(1) Compute the discretized trajectory from initial to final state using the baseline APF.

(2) Use the designated horizon, h, to determine the intermediate goal state from the current discretized APF trajectory.

(3) Determine the control sequence to minimize Eq. (4.1) to the intermediate goal.

(4) Implement the control sequence from (3) over the time step, δ.

(5) Repeat steps (1)-(4) while updating the initial state to that of the current state until the vehicle has arrived at the final goal state.

The intent of this method is to use APF guidance to avoid obstacles and ensure convergence while improving performance with respect to a cost function given by Eq. (4.1). However, even the combined RH-APF method is subject to drift for the same reasons outlined in Chapter 3. This means that the intermediate goal states commanded by the APF may not be reachable. As a result, it may be necessary to perform a linear analysis on the system and make corrections to the applied controls to ensure the states required for convergence or collision avoidance are captured.

$$J = \int_{t}^{t+h} dt + \phi_a \left(\vec{x} \left(t + \delta \right), \vec{x}_d \right) \tag{4.1}$$

where:

h is the time horizon

δ is the time step

ϕ_a is the attractive potential

$\vec{x}_d$ is the final goal state

The APF trajectory will provide a baseline trajectory and the intermediate goal states to the receding horizon planner. The RH algorithm will then determine the appropriate control to minimize transit time to the next goal state. This improves

performance in several ways: the RH planner improves performance with respect to the cost function while hopefully remaining significantly more computationally efficient than the OCP and the APF alleviates the RH planner from the burden of accounting for obstacles via inequality constraints.

4.2.1 Effects of Time Horizon and Time Step Variation. First, the hybrid RH-APF guidance algorithm's performance was analyzed with respect to the time horizon and time step. The analysis was performed on the thruster-powered motor-boat example from the previous chapter for the case $U_{max} = 5$ where U_{max} is the maximum span-wise velocity of the stream. The time step was initially fixed and the horizon was gradually increased. This process was repeated for various length time steps. A minimum step of 0.5 seconds was used due to excessive computation times at smaller values. Table 4.2 summarizes the results at each test condition.

Table 4.1: Effects of varying time step and horizon for RH-APF hybrid algorithm

$\delta = 0.5$	t_f (sec)	Time per step (sec)	Total Run Time (sec)
$h = 0.5$	26.6375	0.0976	9.3754
$h = 1.0$	27.8606	0.0839	4.9514
$h = 1.5$	27.9439	0.0990	4.8559
$h = 2.0$	27.2852	0.1195	5.2612
$h = 5.0$	37.0534	0.2015	13.1032
$h = 10.0$	42.9991	0.5593	47.5473
$\delta = 1.0$	t_f (sec)	Time per step (sec)	Total Run Time (sec)
$h = 1.0$	20.8080	0.2288	9.8407
$h = 1.5$	21.4226	0.2169	6.7275
$h = 2.0$	21.0987	0.2449	6.3703
$h = 2.5$	20.6702	0.2461	5.6627
$h = 5.0$	23.5886	0.2968	4.4544
$h = 10.0$	17.6938	0.6318	28.9236
$\delta = 1.5$	t_f (sec)	Time per step (sec)	Total Run Time (sec)
$h = 1.5$	18.2053	0.4561	12.7737
$h = 2.0$	18.6316	0.4616	10.1581
$h = 2.5$	18.2291	0.4601	8.7441
$h = 5.0$	12.9793	0.4871	3.4125
$h = 10.0$	21.0839	0.8044	10.4599

The horizon, h, had the greatest effect on the final time and total computation time of the algorithm. As h increased, the initial trend was for computation time to drop. However, as h became much larger than δ, computation time began to increase drastically. The latter trend was expected given that as h increases, the fixed horizon optimal control problem is approached. The tendency for computation to increase as h approaches δ may seem unusual at first glance, but it is easily explainable. As the time horizon decreases, the number of calls to the optimization subroutine increases. Additionally, there is a minimum computation time associated with each run of the optimizer. This means that there exists some value of h that is greater than δ such that total computation time is minimized. The same trend was observed for final time. This may seem counter intuitive because letting h approach t_f would represent the optimal solution. However, since the optimizer does not have full knowledge of the environment, smaller horizons actually provide a better solution since more corrections can be made following propagation of the equations of motion. Figures 4.2, 4.3, and 4.4 show the trajectory variation for a sampling of the horizon and time step settings.

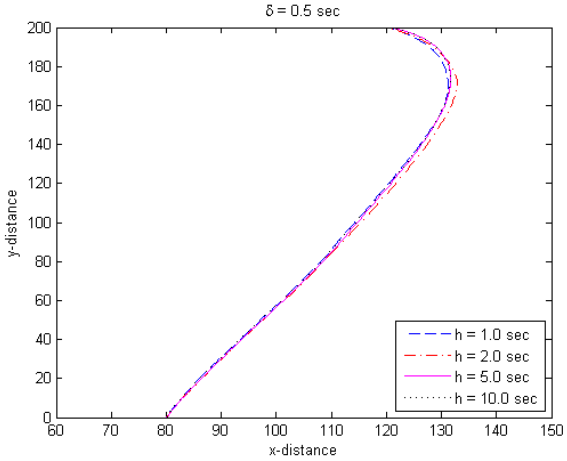

Figure 4.2: Trajectory comparison, $\delta = 0.5$ sec

Varying the time step size mostly influenced the average computation time per step. Smaller time steps were associated with smaller computation time per step. This measure of performance is very important when considering real-time guidance

applications. The maximum allowable time per step would be directly related to the update rate of the guidance system. This indicates the need for a sensitivity analysis for each unique application of the hybrid guidance method to determine the most suitable time horizon and time step. The remaining analysis will be performed assuming that the guidance system has a refresh rate of 10 Hz. Therefore, based on the tabulated results and a 10 Hz refresh rate, the time horizon and time step combination chosen for the remaining analysis was $h = 1.5$ s and $\delta = 0.5$ s.

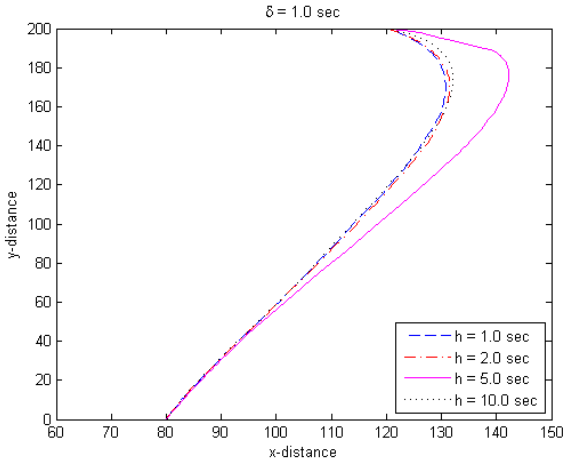

Figure 4.3: Trajectory comparison, $\delta = 1.0$ sec

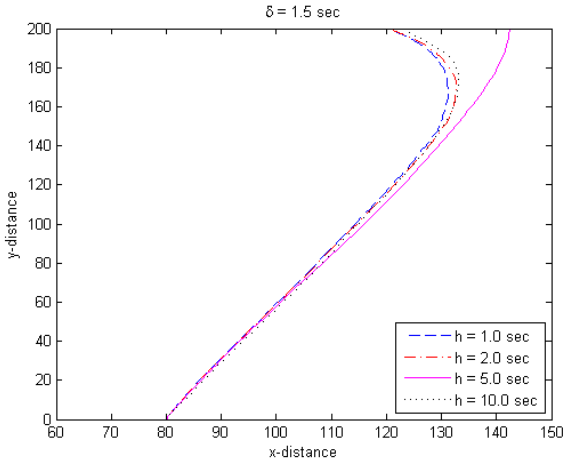

Figure 4.4: Trajectory comparison, $\delta = 1.5$ sec

4.2.2 Results from Saturated Motorboat Case Study. The analysis of the hybrid RH-APF guidance algorithm was continued for the saturated motorboat case

study at the same conditions as the parametric study of Section 3.2.2 to determine whether the hybrid algorithm was in fact an improvement on the results previously obtained. The settings of the APF portion of the hybrid algorithm were unchanged from the Chapter 3 methodology. The receding horizon planner was run with a horizon and step of 1.5 and 0.5 seconds respectively. For consistency, the Δt was set at 0.1 seconds. Figure 4.5 shows the results plotted alongside the Chapter 3 data from the saturated case study for comparison. The unsaturated case study was forgone since the results would exactly equal the baseline quadratic APF performance. This was because in order to minimize time, the optimizer will drive the controls to infinity and lead to an infeasible solution.

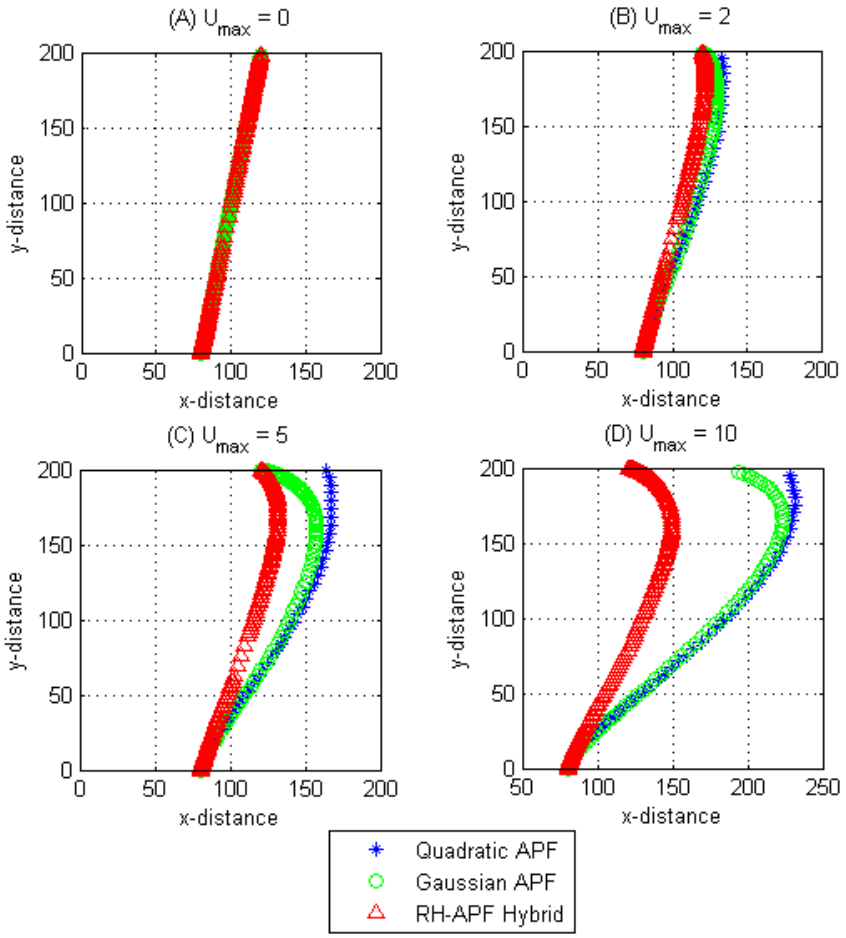

Figure 4.5: Trajectory comparison of hybrid and baseline APF methods

As indicated by figure 4.5, the biggest improvement over both baseline APF algorithms was, surprisingly, the convergence to the goal state in both position and velocity. Some improvement was anticipated due to the optimization occurring within the RH subroutine, but the drastic difference from the baseline APF convergence was not expected. Based on the differences evident in the Chapter 3 saturated and unsaturated figures, it appears as if the RH portion of the hybrid planner has mitigated some of the effects of saturation. In the interest of due diligence, a thorough review of the algorithm was performed with particular attention paid to the saturation settings. While no discrepancies were discovered, the possibility of an error within the algorithm exists as an alternative explanation. Table 4.2 compares overall performance between the algorithms to include run time and total cost. Not surprisingly, the hybrid algorithm took longer to compute than the APF due to the receding horizon planning subroutine. In terms of cost (transit time), the hybrid greatly outperformed the baseline APFs for the $U_{max} = 0$ case. For the remaining cases, the final time fell between the Gaussian and quadratic APF performance, but more closely to the quadratic.

Table 4.2: Comparison of hybrid algorithm performance with baseline APF data for saturated motorboat case study

Saturated Motorboat					
Case	APF Type	t_f (sec)	Total Distance	x-distance from Goal	Total Run Time (sec)
$U_{max} = 0$	Gaussian	204.3	203.96	0	0.3115
	Quadratic	204.2	203.96	0	0.5765
	Hybrid	26.3632	203.96	0	4.6412
$U_{max} = 2$	Gaussian	55.3	210.50	0.01	0.0771
	Quadratic	22.6	209.86	13.42	0.1224
	Hybrid	27.4274	205.25	0	4.6703
$U_{max} = 5$	Gaussian	55.1	239.04	0.03	0.1084
	Quadratic	22.1	223.21	13.47	0.1239
	Hybrid	27.9439	211.48	0	4.9206
$U_{max} = 10$	Gaussian	55.1	276.33	185.60	0.1102
	Quadratic	21.9	256.59	106.84	0.1091
	Hybrid	32.5071	229.44	0	4.9908

4.2.3 Results from Sailboat Case Study. Like the motorboat, the sailboat trajectory was also improved by implementing the RH-APF guidance algorithm. However, due to the indirect control of the sailboat, convergence remained problematic. Figure 4.6 shows the comparison between the trajectories generated by baseline APF and the RH-APF.

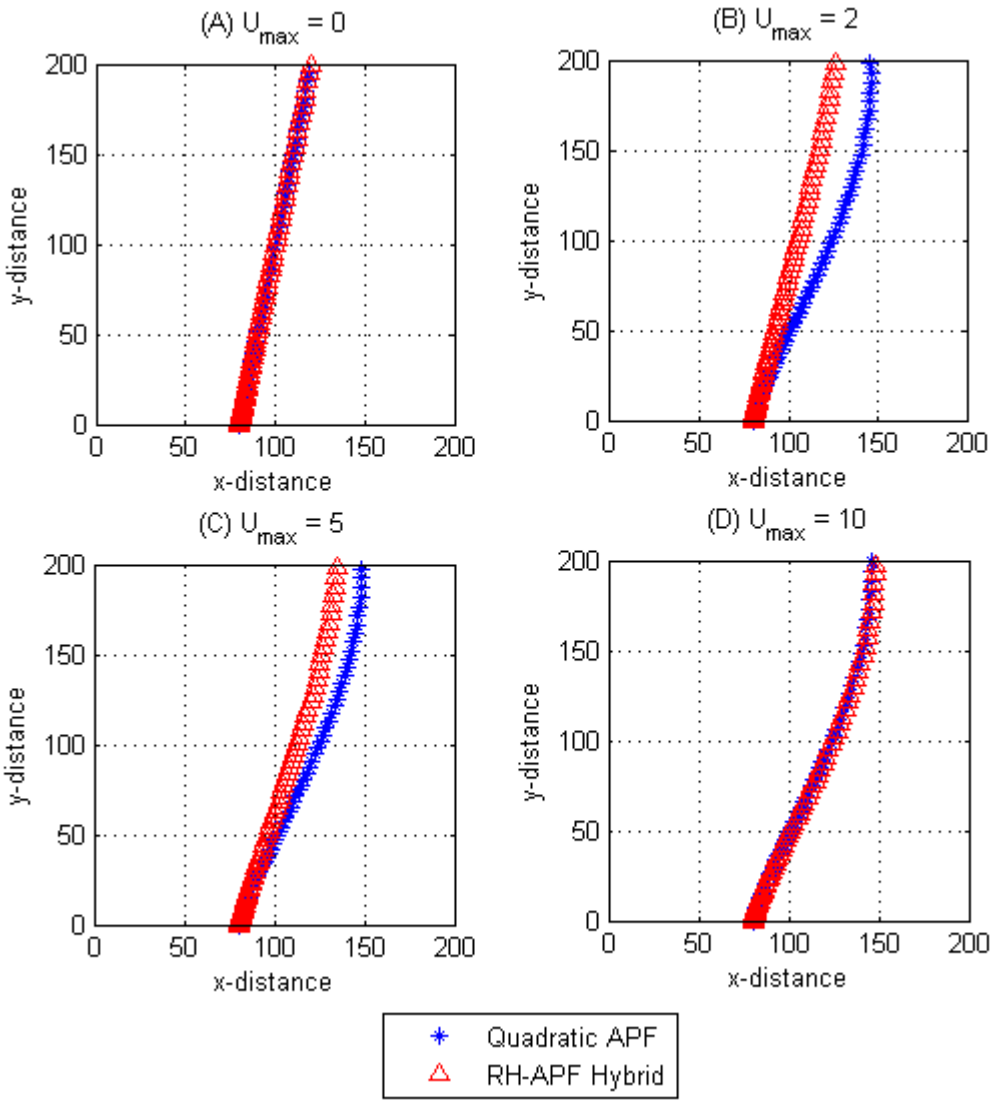

Figure 4.6: Trajectory comparison of hybrid and baseline APF methods for sailboat case study

As expected, the hybrid algorithm had larger total computation times for each case due to the optimization subroutine. However, the computation time per step was not greatly increased. While the hybrid algorithm did not converge to the goal state, it did manage to get closer to the goal than the baseline APF in each case. As evident in figure 4.6, performance of the hybrid algorithm began to degrade and approach the baseline APF as the stream velocity neared $U_{max} = 10$. The deceivingly small final transit times recorded for each algorithm in Table 4.3 are a result of failure to converge on the desired final velocity. In other words, due to the saturation induced by the indirect control, the sailboat cannot adequately decelerate as the goal is approached.

Table 4.3: Comparison of hybrid algorithm performance with baseline APF data for sailboat case study

Sailboat Case Study					
Case	APF Type	t_f (sec)	Total Distance	x-distance from Goal	Total Run Time (sec)
$U_{max} = 0$	Quadratic	8.2	203.96	0	0.0788
	Hybrid	10.5	203.96	0	6.9477
$U_{max} = 2$	Quadratic	8.3	221.08	6.8525	1.3158
	Hybrid	10.5	204.49	5.7783	7.2534
$U_{max} = 5$	Quadratic	8.1	249.94	15.8611	1.3222
	Hybrid	10.0	211.03	13.9908	7.2611
$U_{max} = 10$	Quadratic	8.0	254.17	28.1262	1.3823
	Hybrid	10.0	221.08	27.8583	7.9648

4.3 Summary

The results show marked, albeit surprising, improvement in all measures of performance when the RH-APF algorithm was implemented. Total cost was greatly decreased and computation time per step could be managed by the appropriate selection of time horizon and time step. While these results are promising, the unexpected convergence of the saturated case studies remain highly suspect. Further review of the hybrid algorithm is recommended to trace possible errors in implementation.

The RH-APF sailboat results matched more closely with expectations. Convergence remained an issue due to the reachability of the desired states. Because of

the limited control authority exacerbated by the indirect control, the required force to reach a commanded state was often unattainable. Not only does this prevent the vehicle from reaching the desired goal, but obstacle avoidance is no longer guaranteed. Recognizing this limitation, future work on the hybrid algorithm would benefit from determining in advance whether a commanded state can be reached and use the results to shape the APF such that convergence and obstacle avoidance are possible.

V. Conclusions

The primary objective of this research was to examine APF performance when applied to systems with limited control authority and external dynamics and use the findings to develop a hybrid guidance algorithm that balanced computational efficiency and optimality. This was done by implementing APF and optimal control guidance schemes on simplified case studies.

The research confirmed that not only was APF guidance lacking in performance with respect to a cost function, but it could not guarantee convergence while operating in dynamic environments. Additionally, it was shown that due to high computation times, optimal control can be infeasible when a high refresh rate is required. However, performance with respect to both computation time and cost was improved by hybridizing the APF approach with receding horizon planning. Results showed that for the RH-APF hybrid, computation time was improved from the optimal control solution while improving the convergence and cost from the baseline APF solution.

While the hybrid method greatly improved performance for a saturated system in dynamic environment, this was limited to the direct control case. Slight improvements were seen for the indirect control within a dynamic environment, but convergence could not be guaranteed with the current state of the algorithm. Based on this initial data, the hybrid approach shows promise in regard to implementation within a real-time guidance scheme, however, it must be further vetted before its effectiveness can be guaranteed.

The secondary objective of this research was to determine what classes of problems are well-suited to APFs or APF-hybrids. The analysis performed suggests that the utility of artificial potential functions alone can be highly case specific. For example, the APF guidance algorithm performed the best when applied to a fully actuated system with limited saturation. Additionally, when subjected to external dynamics, the type of the dynamics can greatly effect the success of the APF guidance algorithm. Initial data shows that APF guidance is best supplemented with alternative guidance methods, such as receding horizon optimal control planning, to ensure guarantees can

be made on performance with respect to both cost functions and convergence. This was shown to be especially true when the vehicle of interest was subjected to external forces as well as control authority limitations.

While this work investigated the utility of various guidance methods with respect to minimum transit time problems, results may vary by exploring alternative cost functions such as minimum control effort, minimum distance traveled, etc. By doing so, it may be evident that the guidance methods in question are better suited toward a specific class of optimization problems. Additionally, there is work to be done in order to improve the RH-APF hybrid for systems with indirect control. A potential solution to this problem includes linearization of the system dynamics in order to perform a reachability analysis. Specifically, it may be useful to locally linearize the system dynamics at each time step of the algorithm to estimate if a commanded state may be reached. If the linear analysis predicts the state to be unreachable, then it would be desirable to use some measure of the degree to which a state cannot be reached in order to apply a correction within the guidance algorithm. The author would like to investigate using such a measure to shape the attractive potential as a function of time.

Appendix A. Sailboat Equations of Motion Derivation

This appendix presents the derivation of the discretized equations of motion for the sailboat.

Assuming acceleration is directly proportional to the wind velocity and drag, D,

$$\dot{V}_{sw} = V_w cos\left(\theta - \beta\right) - D \tag{A.1}$$

Where:

V_{sw} is the sailboat velocity due to the wind

V_w is the wind velocity

θ is the sail direction

β is the wind direction

D is the drag force

The controls to be found are sail direction and drag. Integrating Eq. (A.1) across the time step, with θ, D, and the wind parameters assumed constant over the time step,

$$\int_{V_{sw}(0)}^{V_{sw}(\Delta T)} dV_{sw} = \int_{0}^{\Delta T} \left\{V_w \cos\left(\theta - \beta\right) - D\right\} dT \tag{A.2}$$

$$V_{sw}\left(\Delta T\right) = V_{sw}\left(0\right) + \Delta T \left\{V_w \cos\left(\theta - \beta\right) - D\right\} \tag{A.3}$$

The total stream velocity, V_R, is described by the following relationship:

$$V_R = 4\frac{V_{RM}}{h^2} y\left(\Delta T\right)\left(h - y\left(\Delta T\right)\right) \tag{A.4}$$

Where:

V_{RM} is the peak parabolic stream velocity

h is the fixed stream width

$y\left(\Delta T\right)$ is the y-position at time at time ΔT

The velocity in the y-direction is described by Eq. (A.5).

$$\dot{y}\left(\Delta T\right) = v_{sw}\left(\Delta T\right)\sin\,\theta \tag{A.5}$$

Where:

v_{sw} is the y-component of the sailboat velocity due to the wind

Substituting Eq. (A.3) into Eq. (A.5) and integrating,

$$\int_{y(0)}^{y(\Delta T)} dy = \int_{0}^{\Delta T} \left\{v_{sw}\left(0\right) + [V_w \cos\left(\theta - \beta\right) - D_y]\Delta T\right\}\sin\,\theta\; dT \tag{A.6}$$

$$y\left(\Delta T\right) = y\left(0\right) + \Delta T\left\{v_{sw}\left(0\right) + \frac{\Delta T}{2}[V_w \cos\left(\theta - \beta\right) - D_y]\sin\,\theta\right\} \tag{A.7}$$

Similarly, in the x-direction:

$$\dot{x} = V_R + V_{sw} cos\ \theta \tag{A.8}$$

Substituting the expressions for V_R and V_{sw} and separating variables, the following equation can be integrated. For brevity, the expression for $y(\Delta T)$ was not substituted into Eq. (A.9) but was accounted for during actual integration.

$$\int_{x(0)}^{x(\Delta T)} dx = \int_{0}^{\Delta T} \left\{ 4\frac{V_{RM}}{h^2} y(\Delta T)(h - y(\Delta T)) + V_{sw}(0) + \Delta T[V_w \cos(\theta - \beta) - D] \right\} dT \tag{A.9}$$

Equation (A.10) is the final expression for x-position.

$$\begin{aligned} x(\Delta T) =& x(0) + \Delta T\{u_{sw}(0) + C(0)\,\mathrm{cosT}(0)\} + \frac{4V_{RM}}{h^2}\left\{ \frac{\Delta T^4}{4}[D_y(0)\, v_{sw}(0) \right. \\ & - C(0)\,\mathrm{sinT}(0)\, v_{sw}(0)] + h[\frac{\Delta T^3}{6}(C(0)\,\mathrm{sinT}(0) - D_y(0)) \\ & + \frac{v_{sw}(0)\,\Delta T^2}{2} + y(0)\,\Delta T] - \frac{\Delta T^3}{3}[v_{sw}^2(0) - D_y(0)\, y(0) + C(0)\,\mathrm{sinT}(0)\, y(0)] \\ & - \Delta T y^2(0) - \frac{\Delta T^5}{20}[C^2(0)\,\mathrm{sinT}^2(0) - 2C(0)\, D_y(0)\,\mathrm{sinT}(0) + D_y^2(0)] \\ & \left. - \Delta T^2 v_{sw}(0)\, y(0) \right. \end{aligned} \tag{A.10}$$

Continued

Listing B.1: Quadratic APF for Motorboat

```
function [r,v,dv,theta,grad_pot]=sailboatAPF_quad

clear;clc;close all
h = 200;    % Stream width
d =  40;    % x-dist to target
U_max =  10;% Max stream velocity (center)
Beta  = 130;% Wind direction (pos. CCW from x-axis)
W_max =  10;% Wind speed
W = [W_max*cosd(Beta);W_max*sind(Beta)];
x1 = h/2 - d/2;
x2 = h/2 + d/2;
A = [x1; 0];% Starting point, A
B = [x2; h];% Ending point, B

% Sandbar (Gaussian velocity distribution over ellipse)
sbar.xc = [80 100 100]; % X-position of ellipse center(s)
sbar.yc = [40  90 140]; % Y-position of ellipse center(s)
sbar.a  = [20  30  30]; % Ellipse semi-major axes
sbar.b  = [10  15  10]; % Ellipse semi-minor axes
sbar.Vsmax = 0*[.2*U_max .25*U_max .1*U_max];
    % Note that this velocity adds to U_max.
    % If a total velocity is specified be sure to subtract U_max
% Obstacles
obst.xc = [100 110 110 120 120 120];
obst.yc = [ 25  50  75 100 125 150];
obst.rad= 0*[ 20  30  20  20 20  20];
obst.mu = [1/5 1/5 1/5 1/5 1/10 1/10];
%% APF parameters
Q1 = 1/500*eye(2);   % position     % attractive potential ...
   weighting matrix
Q2 = 0/100*eye(2);   % velocity
mu = 1/5;            % repulsive potential
```

```
pot_ang = .00001;
K =100*[1 0;0 1];              % gain matrix

%% Simulation
dt = 0.1;
dv_max = 1;
r(:,1) = A;
v(:,1) = [0,0];
theta(:,1) = 130;
Psi(:,1) = [cosd(theta);sind(theta)];
U(:,1) = vel_prof(A,U_max,h,sbar);
start = tic;
i = 2;
while abs(r(2,i-1)-h)>0.1 && r(2,i-1)<h
    inner = tic;
    grad_pot(:,i-1) = att_grad_pot(r(:,i-1)-B,v(:,i-1),Q1,Q2)+...
        rep_grad_pot(r(:,i-1),Q1,obst);
    if norm(v(:,i-1))≠0
    ang(i) = ...
       acosd(dot(v(:,i-1)/norm(v(:,i-1)),(-grad_pot(:,i-1))/...
        norm(grad_pot(:,i-1))));
    else
    ang(i) = acosd(dot(v(:,i-1),(-grad_pot(:,i-1))));
    end
    if ang(i) > pot_ang
        dv(:,i) = -K*grad_pot(:,i-1)-v(:,i-1);
        if abs(norm(dv(:,i)))>dv_max  % saturation alg.
            dv(:,i) = dv_max*dv(:,i)/norm(dv(:,i));
        end
    else
        dv(:,i) = [0;0];
    end
[r(:,i),v(:,i),r_apf(:,i)] = ...
   EOM(r(:,i-1),v(:,i-1),dv(:,i),Beta,W_max,...
    U_max,h,sbar,dt);
```

```
        elapsed_time(i-1) = toc(inner);
66      i = i+1;

    end
    toc(start)
    h1  = openfig('ocp.fig','reuse'); % uncomment if OCP run @ ...
       same conditions
71  ax1 = gca;
    fig1 = get(ax1,'children');
    xdat_ocp  = get(fig1,'xdata');
    ydat_ocp  = get(fig1,'ydata');
    h2 = figure;
76  s1 = subplot(1,2,1);
    copyobj(fig1,s1); hold on; box on
    colormap cool
    t = 0:dt:(length(r)-2)*dt;
        plot(r(1,1:(end-1)),r(2,1:(end-1)),'b.')
81      axis([0 h 0 h]);axis square
        xlabel('x-distance');ylabel('y-distance')
    subplot(1,2,2)
        [X,Y] = meshgrid(linspace(0,200,50),linspace(0,200,50));
        for k = 1:50;
86          for j = 1:50;
                pfield(k,j) = 0.5*[X(k,j)-B(1); Y(k,j)-B(2)]'*Q1*...
                    [X(k,j)-B(1);Y(k,j)-B(2)];
            end
        end
91      contour(X,Y,pfield);hold on
        colormap cool
        plot(xdat_ocp{1,1},ydat_ocp{1,1},'ko','markerfacecolor','k')
        plot(r(1,1:(end-1)),r(2,1:(end-1)),'b.')
        axis([0 h 0 h]); axis square
96      xlabel('x-distance');

            figure
```

```
        n = 20;
        clear X Y pfield
101     [X,Y] = meshgrid(linspace(0,h,n),linspace(0,h,n));
        for k = 1:n;
            for j = 1:n;
                pfield(k,j) = [X(k,j)-B(1); ...
                    Y(k,j)-B(2)]'*Q1*[X(k,j)-...
                     B(1);Y(k,j)-B(2)];
106         end
        end
        contour(X,Y,pfield,n);hold on
        colormap cool

111     for k = 1:n
        d = [X(k,:);Y(k,:)];
        for j = 1:n
            P = -B+d(:,j);
        grad(:,j)= Q1*P; % add repulsive potential
116     end
        quiver(d(1,:),d(2,:),-K(1,1)*grad(1,:),-K(2,2)*grad(2,:),...
            0.25,'color',[.1 .7 .9]) ;hold on
    end
      plot(r(1,1:(end-1)),r(2,1:(end-1)),'b.',r_apf(1,2:(end-1)),...
121       r_apf(2,2:(end-1)),'r^')
        axis([0 h 0 h]); axis square
        xlabel('x-distance');ylabel('y-distance')
        str = ['U_{max} = ',num2str(U_max)];
        title(str)
126
    figure
    open ocp2.fig % uncomment if OCP solution has been run @ same ...
       conditions
        subplot(4,1,1)
    %     plot(t,theta(1:(end-1)),'b.');hold on;xlim([0 t(end)])
131     ylabel('\theta (t)')
```

```
        subplot(4,1,2)
        plot(t,sqrt(sum(v(:,1:(end-1)).^2,1)),'b.');hold ...
            on;xlim([0 t(end)])
        ylabel('V(t)')
        subplot(4,1,3)
        plot(t,r(1,1:(end-1)),'b.');hold on;xlim([0 t(end)])
        ylabel('x(t)')
        subplot(4,1,4)
        plot(t,r(2,1:(end-1)),'b.');hold on;xlim([0 t(end)])
        xlabel('time');ylabel('y(t)')
    end

    %% Attractive potential
    function grad = att_grad_pot(r,v,Q1,Q2)  %grad of attractive ...
        potential

    % Quadratic potential function
    grad = Q1*r + Q2*v; % position and velocity goal state
    end
    %% Repulsive potential
    function grad = rep_grad_pot(r,Q,obst)

    for i = 1:length(obst.xc)
        r_obs = [obst.xc(i);obst.yc(i)];
        P_inv = 1/obst.rad(i)^2*eye(2);
        if obst.rad(i)==0
            grad(:,i)=[0;0];
        else
            grad(:,i) = ...
                obst.mu(i)/((r-r_obs)'*P_inv*(r-r_obs)-1)*(eye(2) - ...
                 P_inv*(r-r_obs)*r'/((r-r_obs)'*P_inv*(r-r_obs)-1))*Q*r;
        end
    end
    grad = sum(grad,2);
    end
```

```
%% Velocity profile
function U = vel_prof(r,U_max,h,sbar)

x = r(1);
y = r(2);

U = 4*U_max*y.*(h-y)./(h^2);
xc = sbar.xc;
yc = sbar.yc;
sig_x  = sbar.a;
sig_y  = sbar.b;
Vsmax = sbar.Vsmax;
Vs = 0;
for k = 1:length(xc)
temp = ...
   Vsmax(k).*exp(-((x-xc(k)).^2)/(2*sig_x(k)^2)-((y-yc(k)).^2)/...
    (2*sig_y(k)^2));
Vs = Vs+ temp;
clear temp
end
U = [U+Vs;0];
end
%% Equations of motion
% function V = EOM(r,theta,Beta,W_max,U_max,h,sbar)

function [R,V,R_apf] = EOM(r,vel,dv,Beta,W_max,Umax,h,sbar,dT)

x0 = r(1);
y0 = r(2);
u0 = vel(1);
v0 = vel(2);
Dy = 0; % for now

u = u0+dv(1);
v = v0+dv(2);
```

```
        x = ...
            x0+dT*(u0+(dT/2)*dv(1)/dT)+(4*Umax/(h^2))*(dT^4*((Dy*v0)/4 ...
            ...
             -(dv(2)/dT*v0)/4)+h*(((dv(2)/dT)/6-Dy/6)*dT^3+(v0*dT^2)/2+...
201          y0*dT)-dT^3*(v0^2/3-(Dy*y0)/3+(dv(2)/dT*y0)/3)-dT*y0^2-...
             dT^5*((dv(2)^2/dT^2)/20 ...
                 -(Dy*dv(2)/dT)/10+Dy^2/20)-dT^2*v0*y0);
        y = y0 + dT*(v0+(dT/2)*(dv(2)/dT-Dy));
    R = [x;y];
    R_apf = [x0+u*dT;y0+v*dT];
206 V = [u;v];
    end
```

Listing B.2: Quadratic APF for Sailboat

```
function [r,v,dv,theta,grad_pot]=sailboatAPF_quad_theta

clear;clc;close all
h = 200;     % Stream width
d =  40;     % x-dist to target
U_max =   0;% Max stream velocity (center)
Beta  = 130;% Wind direction (pos. CCW from x-axis)
W_max =  10;% Wind speed
W = [W_max*cosd(Beta);W_max*sind(Beta)];
x1 = h/2 - d/2;
x2 = h/2 + d/2;
A = [x1; 0];% Starting point, A
B = [x2; h];% Ending point, B

% Sandbar (Gaussian velocity distribution over ellipse)
sbar.xc = [80 100 100]; % X-position of ellipse center(s)
sbar.yc = [40  90 140]; % Y-position of ellipse center(s)
sbar.a  = [20  30  30]; % Ellipse semi-major axes
sbar.b  = [10  15  10]; % Ellipse semi-minor axes
sbar.Vsmax = [.2*U_max .25*U_max .1*U_max];
    % Note that this velocity adds to U_max.
    % If a total velocity is specified be sure to subtract U_max
% Obstacles
obst.xc = [ 85];
obst.yc = [100];
obst.rad= [0];

%% APF parameters
Q1 = 1/500*eye(2);   % attractive potential weighting matrix
mu = 1/5;            % repulsive potential
pot_ang = .00001;
K =1*[1 0;0 1];             % gain matrix

%% Simulation
```

```
dt = 0.1;%.5;
r(:,1) = A;
v(:,1) = [0,0];
theta(:,1) = 130;
Psi(:,1) = [cosd(theta);sind(theta)];
U(:,1) = vel_prof(A,U_max,h,sbar);
start = tic;
i = 2;
while abs(r(2,i-1)-h)>0.1 && r(2,i-1)<h
    inner = tic;
    grad_pot(:,i-1) = att_grad_pot(r(:,i-1)-B,Q1)+...
                      rep_grad_pot(r(:,i-1),Q1,mu,obst);
    if norm(v(:,i-1))≠0
    ang(i) = ...
       acosd(dot(v(:,i-1)/norm(v(:,i-1)),(-grad_pot(:,i-1))/...
        norm(grad_pot(:,i-1))));
    else
    ang(i) = acosd(dot(v(:,i-1),(-grad_pot(:,i-1))));
    end
    if ang(i) > pot_ang
        dv(:,i) = -K*grad_pot(:,i-1)-v(:,i-1);
        R(:,i)  =  r(:,i-1) + (v(:,i-1)+dv(:,i)).*dt;
        U(:,i)  = vel_prof(R(:,i),U_max,h,sbar);
        Psi0 = Psi(:,i-1);
        [Psi(:,i),¬,flag] = ...
           fsolve(@(Psi0)psisolve(W_max,Beta,v(:,i-1)+...
            dv(:,i),Psi0,dt),Psi0);
        if flag == 1
            theta(:,i) = atan2d(Psi(2,i),Psi(1,i));
        else
            Psi(:,i)   = ...
               (v(:,i-1)+dv(:,i))./norm((v(:,i-1)+dv(:,i)));
            theta(:,i) = atan2d(Psi(2,i),Psi(1,i));
        end
    else
```

```
            dv(:,i) = [0;0];
68          Psi(:,i) = Psi(:,i-1);
            theta(:,i) = theta(:,i-1);
        end
        [r(:,i),v(:,i)] = ...
           EOM(r(:,i-1),v(:,i-1),theta(:,i),Beta,W_max,...
            U_max,h,sbar,dt);
73      r_apf(:,i) = r(:,i-1)+dt*(v(:,i-1)+dv(:,i));
        elapsed_time(i-1) = toc(inner);
        i = i+1;
    end
    total_time = toc(start)
78  % Plotting
    h1  = openfig('dumbocp.fig','reuse');
    ax1 = gca;
    fig1 = get(ax1,'children');
    xdat_ocp  = get(fig1,'xdata');
83  ydat_ocp  = get(fig1,'ydata');
    figure
    subplot(1,2,1)
    [x,y,V_prof]=sailboat_velocity_prof_gauss(sbar,U_max,h);
    [C,hc]=contour(x,y,V_prof,15);hold on
88  vec = 0:1.5:(U_max+max(sbar.Vsmax));
    set(hc,'levellist',vec,'textlist',vec,'showtext','on','labelspacing',...
        250,'textlist',0:3:(U_max+max(sbar.Vsmax)))
    t = 0:dt:(length(r)-2)*dt;
    colormap cool
93      plot(xdat_ocp{2,1},ydat_ocp{2,1},'ko','markerfacecolor','k')
        plot(r(1,1:(end-1)),r(2,1:(end-1)),'b.')
        axis([0 h 0 h]); axis square
        xlabel('x-distance');ylabel('y-distance')
        legend('Velocity Contours','Position (OCP)','Position ...
           (APF)',...
98          'location','northwest')
    subplot(1,2,2)
```

```
[X,Y] = meshgrid(linspace(0,200,50),linspace(0,200,50));
for k = 1:50;
    for j = 1:50;
        pfield(k,j) = 0.5*[X(k,j)-B(1); ...
            Y(k,j)-B(2)]'*Q1*[X(k,j)-...
             B(1);Y(k,j)-B(2)];
    end
end
contour(X,Y,pfield);hold on
colormap cool
plot(xdat_ocp{2,1},ydat_ocp{2,1},'ko','markerfacecolor','k')
plot(r(1,1:(end-1)),r(2,1:(end-1)),'b.')
axis([0 h 0 h]); axis square
xlabel('x-distance');
legend('APF Gradient Contours','Position (OCP)','Position ...
   (APF)',...
    'location','northwest')
    figure
n = 20;
clear X Y pfield
[X,Y] = meshgrid(linspace(0,h,n),linspace(0,h,n));
for k = 1:n;
    for j = 1:n;
        pfield(k,j) = [X(k,j)-B(1); ...
            Y(k,j)-B(2)]'*Q1*[X(k,j)-...
             B(1);Y(k,j)-B(2)];
    end
end
contour(X,Y,pfield,n);hold on
colormap cool

for k = 1:n
d = [X(k,:);Y(k,:)];
for j = 1:n
    P = -B+d(:,j);
```

```
        grad(:,j)= Q1*P; % add repulsive potential
133     end
        quiver(d(1,:),d(2,:),-K(1,1)*grad(1,:),-K(2,2)*grad(2,:),...
            0.25,'color',[.1 .7 .9]) ;hold on
    end
      plot(r(1,1:(end-1)),r(2,1:(end-1)),'b.',r_apf(1,2:(end-1)),...
138       r_apf(2,2:(end-1)),'r^')
        axis([0 h 0 h]); axis square
        xlabel('x-distance');ylabel('y-distance')
        str = ['U_{max} = ',num2str(U_max)];
        title(str)
143
    figure
    open ocp2.fig % uncomment if OCP solution has been run @ same ...
       conditions
        subplot(4,1,1)
        plot(t,theta(1:(end-1)),'b.');hold on;xlim([0 t(end)])
148     ylabel('\theta (t)')
        subplot(4,1,2)
        plot(t,sqrt(sum(v(:,1:(end-1)).^2,1)),'b.');hold ...
           on;xlim([0 t(end)])
        ylabel('V(t)')
        subplot(4,1,3)
153     plot(t,r(1,1:(end-1)),'b.');hold on;xlim([0 t(end)])
        ylabel('x(t)')
        subplot(4,1,4)
        plot(t,r(2,1:(end-1)),'b.');hold on;xlim([0 t(end)])
        xlabel('time');ylabel('y(t)')
158 end

    %% Attractive potential
    function grad = att_grad_pot(r,Q1)  %grad of attractive potential

163 % Quadratic potential function
    grad = Q1*r;
```

```
    end
    %% Repulsive potential
    function grad = rep_grad_pot(r,Q,mu,obst)
168
    for i = 1:length(obst.xc)
        r_obs = [obst.xc(i);obst.yc(i)];
        P_inv = 1/obst.rad(i)^2*eye(2);
        if obst.rad(i)==0
173         grad(:,i)=[0;0];
        else
            grad(:,i) = mu/((r-r_obs)'*P_inv*(r-r_obs)-1)*(eye(2) ...
               - ...
                P_inv*(r-r_obs)*r'/((r-r_obs)'*P_inv*(r-r_obs)-1))*Q*r;
        end
178 end
    grad = sum(grad,2);
    end
    %% Velocity profile
    function U = vel_prof(r,U_max,h,sbar)
183
    x = r(1);
    y = r(2);

    U = 4*U_max*y.*(h-y)./(h^2);
188 xc = sbar.xc;
    yc = sbar.yc;
    sig_x  = sbar.a;
    sig_y  = sbar.b;
    Vsmax = sbar.Vsmax;
193 Vs = 0;
    for k = 1:length(xc)
    temp = Vsmax(k).*exp(-((x-xc(k)).^2)/(2*sig_x(k)^2)-...
        ((y-yc(k)).^2)/(2*sig_y(k)^2));
    Vs = Vs+ temp;
198 clear temp
```

```
    end
    U = [U+Vs;0];
    end
    %% Equations of motion
203 function [R,V,R_apf] = EOM(r,vel,theta,Beta,W_max,Umax,h,sbar,dT)

    x0 = r(1);
    y0 = r(2);
    u0 = vel(1);
208 v0 = vel(2);
    Dy = 0; % for now

    C = W_max*cosd(theta-Beta);
    cosT = cosd(theta);
213 sinT = sind(theta);

        u = u0 + dT*(C*cosT);
        v = v0 + dT*(C*sinT-Dy);

218     x = ...
            x0+dT*(u0+(dT/2)*(C*cosT))+(4*Umax/(h^2))*(dT^4*((Dy*v0)/4 ...
            ...
             -(C*sinT*v0)/4)+h*(((C*sinT)/6-Dy/6)*dT^3+(v0*dT^2)/2+y0*dT)...
             -dT^3*(v0^2/3-(Dy*y0)/3+(C*sinT*y0)/3)-dT*y0^2-...
             dT^5*((C^2*sinT^2)/20-(C*Dy*sinT)/10+Dy^2/20)-dT^2*v0*y0);
        y = y0 + dT*(v0+(dT/2)*(C*sinT-Dy));
223
    R = [x;y];
    R_apf = [x0+u*dT;y0+v*dT];
    V = [u;v];

228 end

    function Psi = psisolve(W_max,Beta,vel,Psi0,dT)
```

```
    du = vel(1);
233 dv = vel(2);
    Dy=0;
    theta = atan2d(Psi0(2),Psi0(1));
    C = W_max*cosd(theta-Beta);
    cosT = cosd(theta);
238 sinT = sind(theta);

    Psi = [du - dT*(C*cosT);
           dv - dT*(C*sinT-Dy)];
    end
```

Listing B.3: Continuous Control APF for Motorboat

```
function [r,v,dv,theta,grad_pot]=sailboatAPF_CC

clear;clc;close all
h = 200;    % Stream width
d =  40;    % x-dist to target
U_max =   5;% Max stream velocity (center)
Beta  = 130;% Wind direction (pos. CCW from x-axis)
W_max =  10;% Wind speed
W = [W_max*cosd(Beta);W_max*sind(Beta)];
x1 = h/2 - d/2;
x2 = h/2 + d/2;
A = [x1; 0];% Starting point, A
B = [x2; h];% Ending point, B

% Sandbar (Gaussian velocity distribution over ellipse)
sbar.xc = [80 100 100]; % X-position of ellipse center(s)
sbar.yc = [40  90 140]; % Y-position of ellipse center(s)
sbar.a  = [20  30  30]; % Ellipse semi-major axes
sbar.b  = [10  15  10]; % Ellipse semi-minor axes
sbar.Vsmax = [0*U_max 0*U_max 0*U_max];
    % Note that this velocity adds to U_max.
    % If a total velocity is specified be sure to subtract U_max
obst.xc = [120];
obst.yc = [160];
obst.rad= [10];

%% APF parameters
Q = 1/2.5*eye(2);       % attractive potential weighting matrix
mu = 0;                 % repulsive potential
N = 1/500*eye(2);
pot_ang = .1;
Kd =3;                  % gain matrix

%% Simulation
```

```
dt = .1;
r(:,1) = A;
v(:,1) = [0,0];
theta(:,1) = 40;
Psi(:,1) = [cosd(theta);sind(theta)];
U(:,1) = vel_prof(A,U_max,h,sbar);

i = 2;
while abs(r(2,i-1)-h)≥0.5
    F(:,i) = ...
        cont_contr(r(:,i-1),v(:,i-1),U_max,h,sbar,obst,Kd,Q,mu,N,B);
    v(:,i) = EOM(r(:,i-1),v(:,i-1)+F(:,i)*dt,U_max,h,sbar);
    r(:,i) = r(:,i-1)+dt*v(:,i);
    i = i+1;
    if i >6000 break
    end
end
thr_ang = atan2d(F(2,:),F(1,:));
loc = find(thr_ang<0);
thr_ang(loc) = thr_ang(loc)+360;
%  open ocp.fig
    t = 0:dt:(length(r)-2)*dt;
    figure(1)
    [X,Y] = meshgrid(linspace(0,200,50),linspace(0,200,50));
    for k = 1:50;
        for j = 1:50;
            pfield(k,j) = 0.5*[X(k,j)-B(1); Y(k,j)-B(2)]'*...
                Q*[X(k,j)-B(1);Y(k,j)-B(2)];
        end
    end
    contour(X,Y,pfield);hold on
    colormap cool
    xlabel('x-distance');ylabel('y-distance')
    plot(r(1,1:(end-1)),r(2,1:(end-1)),'b.')
    axis([0 h 0 h])
```

```
        legend('APF Gradient Contours','Position ...
            (APF)','location','northwest')

    figure(2)
        subplot(3,1,1)
        plot(t,sqrt(sum(v(:,1:(end-1)).^2,1)),'b.');hold on
        ylabel('V(t)')
        subplot(3,1,2)
        plot(t,r(1,1:(end-1)),'b.');hold on
        ylabel('x(t)')
        subplot(3,1,3)
        plot(t,r(2,1:(end-1)),'b.');hold on
        xlabel('time');ylabel('y(t)')
    end

    %% Attractive potential
    function grad = att_grad_pot(r,Q)  %grad of attractive potential

    % No change from satellite APF code
        grad = Q*r;
    end
    %% Repulsive potential
    function grad = rep_grad_pot(r,N,mu,obst)
    for i = 1:length(obst.xc)
        if obst.rad(i)>0
        r_obs = r-[obst.xc(i);obst.yc(i)];
        grad(:,i) = -mu*exp(-r_obs'*N*r_obs)*N*r_obs;
        else grad(:,i) = [0;0];
        end
    end
    grad = sum(grad,2);

    end

    %% Hessian
```

```
function H = hess_pot(r,M,obst,N,mu)
H_rep = zeros(2);
for i = 1:length(obst.xc)
    if obst.rad(i)>0
    r_obs = r-[obst.xc(i);obst.yc(i)];
    H_rep = H_rep+mu*exp(-r_obs'*N*r_obs)*(2*N*r_obs*r_obs'*N-N);
    end
end
H_att = M;
H = H_att+H_rep;
end
%% Velocity profile
function U = vel_prof(r,U_max,h,sbar)

x = r(1);
y = r(2);

U = 4*U_max*y.*(h-y)./(h^2);
xc = sbar.xc;
yc = sbar.yc;
sig_x  = sbar.a;
sig_y  = sbar.b;
Vsmax = sbar.Vsmax;
Vs = 0;
for k = 1:length(xc)
temp = ...
    Vsmax(k).*exp(-((x-xc(k)).^2)/(2*sig_x(k)^2)-((y-yc(k)).^2)/...
     (2*sig_y(k)^2));
Vs = Vs+ temp;
clear temp
end
U = [U+Vs;0];
end
%% Equations of motion
function V = EOM(r,dv,U_max,h,sbar)
```

```
x = r(1);
y = r(2);

U = 4*U_max*y.*(h-y)./(h^2);
xc = sbar.xc;
yc = sbar.yc;
sig_x  = sbar.a;
sig_y  = sbar.b;
Vsmax = sbar.Vsmax;
Vs = 0;
for k = 1:length(xc)
temp = ...
   Vsmax(k).*exp(-((x-xc(k)).^2)/(2*sig_x(k)^2)-((y-yc(k)).^2)/...
    (2*sig_y(k)^2));
Vs = Vs+ temp;
clear temp
end

Vx = U+Vs+dv(1); % Stream only provides velocity in x-dir
Vy =    0+dv(2);
V = [Vx;Vy];
end
%% Continuous control law
function F = cont_contr(r,v,U_max,h,sbar,obst,Kd,Q,mu,N,B)

x = r(1);
y = r(2);
dUstr = 4*U_max*v(2)*(h-2*y)/(h^2);
xc = sbar.xc;
yc = sbar.yc;
sig_x  = sbar.a;
sig_y  = sbar.b;
Usmax = sbar.Vsmax;
dUsbar = 0;
```

```
    for k = 1:length(xc)
        temp = ...
            -Usmax(k)*((x-xc(k))*v(1)/(sig_x(k).^2)-(y-yc(k))*v(2)/...
173          (sig_y(k)^2)).*exp(-((x-xc(k)).^2)/(2*sig_x(k)^2)-...
             ((y-yc(k)).^2)/(2*sig_y(k)^2));
        dUsbar = dUsbar + temp;
        clear temp
    end
178 dUx = dUstr + dUsbar;
    dUy = 0;

    H = hess_pot(r,Q,obst,N,mu);
    grad_pot = att_grad_pot(r-B,Q)+rep_grad_pot(r,N,mu,obst);
183
    accel = -H*v-Kd*(grad_pot+v);

    F(1) = -dUx + accel(1);
    F(2) = -dUy + accel(2);
188 F = F';
    end
```

Listing B.4: Optimal Control Main Run File

```
clear;clc;close all
% Required input
h = 200;      % Stream width
d =  40;      % x-dist to target (Δ b/w start and finish x ...
   positions)
U_max = 10;  % Max stream velocity (centerline; not including ...
   sandbar)
Beta  = 130; % Initial wind direction (pos. CCW from x-axis)
W_max = 10;  % Wind speed (knots)
Dx = 0;

% Sandbar (Gaussian velocity distribution over ellipse)
sbar.xc = [80 100 100]; % X-position of ellipse center(s)
sbar.yc = [40  90 140]; % Y-position of ellipse center(s)
sbar.a  = [20  30  30]; % Ellipse semi-major axes
sbar.b  = [10  15  10]; % Ellipse semi-minor axes
sbar.Vsmax = [.2*U_max .25*U_max .1*U_max];
    % Note that this velocity adds to U_max.
    % If a total velocity is specified be sure to subtract ...
       U_max or
    % acceleration will be huge.
obst.xc = [85]; % Obstacle location
obst.yc = [100];
obst.rad= [0];   % Obstacle radius

% Discretized points
N  = 30;

% Use previous solution as initial guess
% (comment out if no data available or changing N)
load z_star.mat
z0 = z_star; clear z_star

% Optional input (oscillatoroy wind speed and freq)
```

```
varB  = 0;   % Variation in beta;
freqB = 0;   % Beta oscillation frequency
varW  = 0;   % Variation in wind speed
freqW = 0;   % Wind speed oscillation frequency

% Start and Finish Points
x1 = h/2-d/2; % Center start point on graph based on dimensions
x2 = h/2+d/2; %
A = [x1  0];  % Starting point, A
B = [x2  h];

% Oscillatory functions
% Create oscillatory wind speed and direction
tBeta = 0:N-1;
if varB≠0
    Beta = Beta+varB*cos(freqB*tBeta);
end
if varW≠0
    W_max= W_max+varW*cos(freqW*tBeta+pi/2);
end

% Optimization
% Initialize optimizer
theta_guess = 110*ones(1,N);
Dy_guess = 5*ones(1,N);
tf_guess = 18;

if ¬exist('z0','var') % Create initial guess if no data loaded
    z0 = [theta_guess Dy_guess tf_guess];
end

%set upper and lower limits on decision input vector (added by ...
   J. Agte)
lb =[(Beta-90)*ones(1,size(theta_guess,2)) ...
   0*ones(1,size(Dy_guess,2)) 0];
```

```
ub =[(Beta+90)*ones(1,size(theta_guess,2)) ...
    2*ones(1,size(Dy_guess,2)) 200];

% Run optimizer
tic
options = ...
    optimset('algorithm','sqp','display','iter','tolcon',1e-6,...
     'MaxFunEvals',100000,'MaxIter',10000);
[z_star,J_star]=fmincon(@sailboat_obj,z0,[],[],[],[],lb,ub,...
                        @(z)sailboat_con_gauss(z,N,A,B,h,U_max,W_max,...
                        Beta,sbar,obst,Dx),...
                        options);
                    toc
save('z_star.mat','z_star')
[g_star,h_star,g_gradDUM,h_gradDUM,...
 x_star,y_star,xd,yd,u_sw_star,v_sw_star]=...
        sailboat_con_gauss(z_star,N,A,B,h,U_max,W_max,Beta,sbar,obst,Dx);
opt_traj = [x_star;y_star];
save('opt_traj.mat','opt_traj')

sprintf('Max inequality constraint = %0.5g\n',max(g_star))
sprintf('Max equality constraint = %0.5g\n',max(h_star))
sprintf('Final time = %1.2f sec\n',z_star(end))

% Get solution
theta_star = z_star(1:N)';
Dy_star = z_star(N+1:2*N)';
tf_star    = z_star(end);
t = 0:tf_star/(N):tf_star;

% Added last control input for completeness (added by J. Agte)
theta_star(N+1) = theta_star(N);
Dy_star(N+1)=Dy_star(N);

% Plot velocity distribution and contours
```

```
    [x,y,V_prof]=sailboat_velocity_prof_gauss(sbar,U_max,h);
    figure(1)
    plot3(x,y,V_prof)
101 figure(2)
    xlabel('x-distance');ylabel('y-distance');zlabel('Velocity')

    [C,hc]=contour(x,y,V_prof,15);hold on
    % clabel(C) % Label contours (kind of crowded...)
106 vec = 0:1.5:(U_max+max(sbar.Vsmax));
    set(hc,'levellist',vec,'textlist',vec,'showtext','on','labelspacing',...
        250,'textlist',0:3:(U_max+max(sbar.Vsmax)))

    colormap cool
111 xlabel('x-distance');ylabel('y-distance')

    plot(x_star',y_star','ko','markerfacecolor','k'); % Plot boat
    a=0:0.01:2*pi;
    for j=1:length(obst.xc);
116    xp=obst.rad(j)*cos(a);
       yp=obst.rad(j)*sin(a);
       plot(obst.xc(j)+xp,obst.yc(j)+yp,'r');
    end

121 axis([0 h 0 h]);
    legend('Velocity Contours','Position (OCP)')
    % Save a copy to plot APF on top
    saveas(gcf,'ocp.fig')
    % Plot control
126 figure(4)
    plot(t,theta_star,'bo')
    xlabel('Time'); ylabel('Sail Angle (deg)')

    % Plot time histories
131 v  = [xd;yd];
        figure(5)
```

```
        subplot(5,1,1)
        plot(t,theta_star,'ko');hold on
        xL = get(gca,'XLim');
136     yL = get(gca,'YLim');
        ylabel('\theta (t)')
        subplot(5,1,2)
        plot(t,Dy_star,'ko');
        ylabel('D_y(t)')
141     subplot(5,1,3)
        plot(t(1:end),sqrt(sum(v.^2,1)),'ko');hold on
        ylabel('V(t)'); xlim(xL)
        subplot(5,1,4)
        plot(t,x_star,'ko');hold on
146     ylabel('x(t)')
        subplot(5,1,5)
        plot(t,y_star,'ko');hold on
        xlabel('time');ylabel('y(t)')
        % Save for APF overplots
151     saveas(gcf,'ocp2.fig')
```

Listing B.5: Optimal Control Objective File

```
function J = sailboat_obj(z)

tf = z(end);
J  = tf;
```

Listing B.6: Optimal Control Constraint File

```
function [g,h,g_grad,h_grad,x,y,xd,yd,u_sw,v_sw] = ...
    sailboat_con_gauss(z,N,A,B,h,U_max,W_max,Beta,sbar,obst,Dx)

% Get states
theta = z(1:N)';
Dy = z(N+1:2*N)';
tf    = z(end);

% Set initial starting point and velocity (added by J. Agte)
x(1) = A(1);
y(1) = A(2);
u_sw(1) = 0;
v_sw(1) = 0;
xd(1)=0;
yd(1)=0;

% Some misc for the following equations (added by J. Agte)
dT = tf/N;
C = W_max.*cosd(theta-Beta);
sinT = sind(theta);
cosT = cosd(theta);
Vrm = U_max;


for i=1:N
u_sw(i+1) = u_sw(i)+dT*(C(i)*cosT(i)-Dx);
```

```
    v_sw(i+1) = v_sw(i)+dT*(C(i)*sinT(i)-Dy(i));
    x(i+1) = ...
        x(i)+dT*(u_sw(i)+(dT/2)*(C(i)*cosT(i)-Dx))+(4*Vrm/(h^2))*...
              (dT^4*((Dy(i)*v_sw(i))/4 -(C(i)*sinT(i)*v_sw(i))/4)+...
               h*(((C(i)*sinT(i))/6-Dy(i)/6)*dT^3+(v_sw(i)*dT^2)/2+y(i)*dT)...
              -dT^3*(v_sw(i)^2/3-(Dy(i)*y(i))/3+(C(i)*sinT(i)*y(i))/3)-...
               dT*y(i)^2-dT^5*((C(i)^2*sinT(i)^2)/20 ...
             -(C(i)*Dy(i)*sinT(i))/10+Dy(i)^2/20)-dT^2*v_sw(i)*y(i));
    y(i+1) = y(i)+dT*(v_sw(i)+(dT/2)*(C(i)*sinT(i)-Dy(i)));
    end

    xd(2:N+1)=diff(x)./(tf/N);
    yd(2:N+1)=diff(y)./(tf/N);
    % Inequality constraints
    g = [];

    h=[ y(end)-B(2);% Must finish at point B
        x(end)-B(1);
        u_sw(end);
        v_sw(end)];
```

Listing B.7: Optimal Control Velocity Profile

```
function [x,y,V_prof]=sailboat_velocity_prof_gauss(sbar,U_max,h)
% Used for plotting the velocity profile and contour map
% Models elliptic sandbar

[x,y]=meshgrid(0:2:h,0:1:h);
% Stream velocity
U = 4*U_max*y.*(h-y)./(h^2);

% Set up sandbar (see "sailboat_con_gauss.m" for detailed ...
   description)
xc =sbar.xc;
yc =sbar.yc;
sig_x = sbar.a;
sig_y = sbar.b;
Vsmax = sbar.Vsmax;
Vs = zeros(size(U));

for k = 1:length(xc)

temp = Vsmax(k).*exp(-((x-xc(k)).^2)/(2*sig_x(k)^2)-...
        ((y-yc(k)).^2)/(2*sig_y(k)^2));
Vs = Vs+ temp;
clear Vspmax loc temp
end
V_prof = U+Vs;
```

Listing B.8: Receding Horizon Main Run for Motorboat

```
function [r_out,v_out,dv_out,elapsed_time]=receding_horizon_thrust

clear;clc;close all
%% Setup
h = 200;     % Stream width
d =  40;     % x-dist to target
U_max =  10;% Max stream velocity (center)
Beta  = 130;% Wind direction (pos. CCW from x-axis)
W_max =  10;% Wind speed
x1 = h/2 - d/2;
x2 = h/2 + d/2;
A = [x1; 0];% Starting point, A
B = [x2; h];% Ending point, B

dt = 0.1; % time step (APF)
H = 15;   % horizon
D = 5;    % step (RH)
time = 0;
% Sandbar (Gaussian velocity distribution over ellipse)
sbar.xc = [80 100 100]; % X-position of ellipse center(s)
sbar.yc = [40  90 140]; % Y-position of ellipse center(s)
sbar.a  = [20  30  30]; % Ellipse semi-major axes
sbar.b  = [10  15  10]; % Ellipse semi-minor axes
sbar.Vsmax = [0*U_max 0*U_max 0*U_max];
    % Note that this velocity adds to U_max.
    % If a total velocity is specified be sure to subtract U_max
% Obstacles
obst.xc = [ 85 60 120 100];
obst.yc = [50  70 120 150];
obst.rad= 0*[10 20 15 10];
mu = [1/5 1/5 1/5 1/5];            % repulsive potential

jj=1;
R = A;
```

```
v0 = [0;0];
dx_used = 0;
dy_used = 0;
dv_used = [];
x_used = A(1);
y_used = A(2);
u_used = 0;
v_used = 0;
%% APF parameters
Q = 1/500*eye(2);  % attractive potential weighting matrix

v0(:,1) = v0;
start = tic;
while abs(R(2,jj)-h)/h ≥ 0.0001 && R(2,jj)<h

%% Compute APF
inner = tic;
[r,v,dv_apf] = APF(v0,A,B,H,U_max,Beta,W_max,sbar,Q,mu,obst,h,dt);

% Get desired position & velocity based on horizon
if length(r)>H
    B_rh = r(:,H);
    uB = v(:,H);
else
    B_rh = r(:,end);
    uB   = v(:,end);
end
uA = v(:,1);

%% Compute RH
N = H;

dx_guess = dx_used(end)*ones(1,N);
dy_guess = dy_used(end)*ones(1,N);
tf_guess = sum((B_rh-A).^2)^0.5/norm(uB-uA);
```

```
70
   z0 = [dx_guess dy_guess tf_guess];
   max_sat = 0.5;
   lb=[-max_sat*ones(1,size(dx_guess,2)) ...
      -max_sat*ones(1,size(dy_guess,2)) 0];
   ub=[max_sat*ones(1,size(dx_guess,2))  ...
      max_sat*ones(1,size(dy_guess,2)) 30];
75 options = ...
      optimset('algorithm','sqp','display','iter','tolcon',1e-6,...
                       'MaxFunEvals',100000,'MaxIter',10000);
   [z_star,J_star,flag]=fmincon(@(z)rh_thrust_obj(z,uA,N,A,B,H,D,Q,mu,obst,...
                               U_max,W_max,Beta,sbar,h),z0,[],[],[],[],lb,ub,...
                               @(z)rh_thrust_con(z,N,A,B_rh,uA,uB,U_max,...
80                             W_max,Beta,sbar,obst,h,D),options);
   %% Implement RHC (Propagate EOMs)
   if flag≠1
   if length(r)≥D
       k = D;
85 else
       k = length(r);
   end
   dv = dv_apf; clear dv_apf
   u_sw(1) = uA(1);
90 v_sw(1) = uA(2);
   x(1) = A(1);
   y(1) = A(2);
   Dy=0;
   for i=1:k
95
   u_sw(i+1) = u_sw(i)+dv(1,i);
   v_sw(i+1) = v_sw(i)+dv(2,i);

   x(i+1) = x(i)+dt*(u_sw(i)+(dt/2)*dv(1,i)/dt)+...
100         (4*U_max/(h^2))*(dt^4*((Dy*v_sw(i))/4-(dv(2,i)/dt*v_sw(i))/4)+...
            h*(((dv(2,i)/dt)/6-Dy/6)*dt^3+(v_sw(i)*dt^2)/2+y(i)*dt)...
```

```
            -dt^3*(v_sw(i)^2/3-(Dy*y(i))/3+(dv(2,i)/dt*y(i))/3)-dt*y(i)^2-...
             dt^5*((dv(2,i)^2/dt^2)/20-(Dy*dv(2,i)/dt)/10+Dy^2/20)-...
             dt^2*v_sw(i)*y(i));
105 y(i+1) = y(i) + dt*(v_sw(i)+(dt/2)*(dv(2,i)/dt-Dy));

    dv_used    = [dv_used dv(:,i)];
    x_used     = [x_used x(i)];
    y_used     = [y_used y(i)];
110 u_used     = [u_used u_sw(i)];
    v_used     = [v_used v_sw(i)];
    end
    time = [time linspace(time(end)+dt,time(end)+dt*D,D)];

115 else
    dx = z_star(1:N);
    dy = z_star(N+1:2*N);
    tf = z_star(end);

120 x(1) = A(1);
    y(1) = A(2);
    u_sw(1) = uA(1);
    v_sw(1) = uA(2);
    dv = [dx;dy];
125 dT = tf/N;

    for i=1:D

    Dy = 0; % for now
130

    u_sw(i+1) = u_sw(i)+dv(1,i);
    v_sw(i+1) = v_sw(i)+dv(2,i);

    x(i+1) = x(i)+dT*(u_sw(i)+(dT/2)*dv(1,i)/dT)+...
135          (4*U_max/(h^2))*(dT^4*((Dy*v_sw(i))/4-(dv(2,i)/dT*v_sw(i))/4)+...
             h*(((dv(2,i)/dT)/6-Dy/6)*dT^3+(v_sw(i)*dT^2)/2+y(i)*dT)...
```

```
            -dT^3*(v_sw(i)^2/3-(Dy*y(i))/3+(dv(2,i)/dT*y(i))/3)-dT*y(i)^2-...
             dT^5*((dv(2,i)^2/dT^2)/20-(Dy*dv(2,i)/dT)/10+Dy^2/20)-...
             dT^2*v_sw(i)*y(i));
y(i+1) = y(i) + dT*(v_sw(i)+(dT/2)*(dv(2,i)/dT-Dy));

    dv_used    = [dv_used dv(:,i)];
    x_used     = [x_used x(i)];
    y_used     = [y_used y(i)];
    u_used     = [u_used u_sw(i)];
    v_used     = [v_used v_sw(i)];
    end
    time = [time linspace(time(end)+dT,time(end)+dT*D,D)];
    end

    clear x y u_sw v_sw
    %% Update APF, Repeat
    A  = [x_used(end);y_used(end)];
    v0 = [u_used(end);v_used(end)];
    elapsed_time(jj) = toc(inner);
    jj = jj+1;
    R(:,jj) = A;

    end
    total_time = toc(start)
    time_per_step = mean(elapsed_time)
    t_final = time(end)
    r_out = [x_used;y_used];
    v_out = [u_used;v_used];
    dv_out = dv_used;
    V = sqrt(v_out(1,:).^2+v_out(2,:).^2);

    plot(x_used,y_used);
    axis([0 200 0 200])
    figure
    plot(V)
```

```
axis square
end
function [r,v,dv] = ...
    APF(v,A,B,H,U_max,Beta,W_max,sbar,Q,mu,obst,h,dt)

%% Simulation
pot_ang = .00001;
K =100*[1 0;0 1];              % gain matrix
dv_max = 0.5;
r(:,1) = A;
i = 2;
U_max=0;
while abs(r(2,i-1)-h)>0.01 && r(2,i-1)<h && i ≤ H
grad_pot(:,i-1) = att_grad_pot(r(:,i-1)-B,Q)+...
                  rep_grad_pot(r(:,i-1),Q,mu,obst);
    if norm(v(:,i-1))≠0
    ang(i) = ...
        acosd(dot(v(:,i-1)/norm(v(:,i-1)),(-grad_pot(:,i-1))/...
         norm(grad_pot(:,i-1))));
    else
    ang(i) = acosd(dot(v(:,i-1),(-grad_pot(:,i-1))));
    end
    if ang(i) > pot_ang
        dv(:,i) = -K*grad_pot(:,i-1)-v(:,i-1);
        if abs(norm(dv(:,i)))>dv_max  % saturation alg.
            dv(:,i) = dv_max*dv(:,i)/norm(dv(:,i));
        end
    else
        dv(:,i) = [0;0];
    end
[r(:,i),v(:,i),r_apf(:,i)] = ...
    EOM(r(:,i-1),v(:,i-1),dv(:,i),Beta,W_max,...
     U_max,h,sbar,dt);
     i = i+1;
end
```

```
    end
%% Attractive potential
function grad = att_grad_pot(r,Q)  %grad of attractive potential

% Quadratic potential function
grad = Q*r; % position and velocity goal state

end
%% Repulsive potential
function grad = rep_grad_pot(r,Q,mu,obst)

for i = 1:length(obst.xc)
    r_obs = [obst.xc(i);obst.yc(i)];
    P_inv = 1/obst.rad(i)^2*eye(2);
    if obst.rad(i)==0
        grad(:,i)=[0;0];
    else
        grad(:,i) = ...
           mu(i)/((r-r_obs)'*P_inv*(r-r_obs)-1)*(eye(2) - ...
            P_inv*(r-r_obs)*r'/((r-r_obs)'*P_inv*(r-r_obs)-1))*Q*r;
    end
end
grad = sum(grad,2);
end
%% Velocity profile
function U = vel_prof(r,U_max,h,sbar)

x = r(1);
y = r(2);

U = 4*U_max*y.*(h-y)./(h^2);
xc = sbar.xc;
yc = sbar.yc;
sig_x  = sbar.a;
sig_y  = sbar.b;
```

```
    Vsmax = sbar.Vsmax;
    Vs = 0;
240 for k = 1:length(xc)
    temp = ...
        Vsmax(k).*exp(-((x-xc(k)).^2)/(2*sig_x(k)^2)-((y-yc(k)).^2)/...
         (2*sig_y(k)^2));
    Vs = Vs+ temp;
    clear temp
245 end
    U = [U+Vs;0];
    end
    %% Equations of motion
    % function V = EOM(r,theta,Beta,W_max,U_max,h,sbar)
250
    function [R,V,R_apf] = EOM(r,vel,dv,Beta,W_max,Umax,h,sbar,dT)

    x0 = r(1);
    y0 = r(2);
255 u0 = vel(1);
    v0 = vel(2);
    Dy = 0; % for now

    u = u0+dv(1);
260 v = v0+dv(2);

    x = x0+dT*(u0+(dT/2)*dv(1)/dT)+(4*Umax/(h^2))*(dT^4*((Dy*v0)/4 ...
       -(dv(2)/dT*v0)/4)+h*(((dv(2)/dT)/6-Dy/6)*dT^3+(v0*dT^2)/2+y0*dT)...
       -dT^3*(v0^2/3-(Dy*y0)/3+(dv(2)/dT*y0)/3)-dT*y0^2-dT^5*((dv(2)^2/dT^2)...
265    /20 -(Dy*dv(2)/dT)/10+Dy^2/20)-dT^2*v0*y0);
    y = y0 + dT*(v0+(dT/2)*(dv(2)/dT-Dy));

    R = [x;y];
    R_apf = [x0+u*dT;y0+v*dT];
270 V = [u;v];
    end
```

Listing B.9: Receding Horizon Objective File for Motorboat

```
function J = ...
    rh_thrust_obj(z,uA,N,A,B,H,D,Q,mu,obst,Umax,W_max,Beta,sbar,h)

dx = z(1:N);
dy = z(N+1:2*N);
tf = z(end);
Umax=0;
x(1) = A(1);
y(1) = A(2);
u_sw(1) = uA(1);
v_sw(1) = uA(2);
dv = [dx;dy];
dT = tf/N;

for i=1:H

Dy = 0; % for now

u_sw(i+1) = u_sw(i)+dv(1,i);
v_sw(i+1) = v_sw(i)+dv(2,i);

x(i+1) = ...
    x(i)+dT*(u_sw(i)+(dT/2)*dv(1,i)/dT)+(4*Umax/(h^2))*(dT^4*...
         ((Dy*v_sw(i))/4-(dv(2,i)/dT*v_sw(i))/4)+...
          h*(((dv(2,i)/dT)/6-Dy/6)*dT^3+(v_sw(i)*dT^2)/2+y(i)*dT)...
         -dT^3*(v_sw(i)^2/3-(Dy*y(i))/3+(dv(2,i)/dT*y(i))/3)-...
          dT*y(i)^2-dT^5*((dv(2,i)^2/dT^2)/20 ...
         -(Dy*dv(2,i)/dT)/10+Dy^2/20)-dT^2*v_sw(i)*y(i));
y(i+1) = y(i) + dT*(v_sw(i)+(dT/2)*(dv(2,i)/dT-Dy));
end

r = [x(D);y(D)]-B;
J = z(end) + att_pot(r,Q);
```

```
end

%% Attractive potential
function ap = att_pot(r,Q)  %grad of attractive potential

% Quadratic potential function
ap = 0.5*r'*Q*r; % position and velocity goal state

end
%% Repulsive potential
function grad = rep_grad_pot(r,Q,mu,obst)

for i = 1:length(obst.xc)
    r_obs = [obst.xc(i);obst.yc(i)];
    P_inv = 1/obst.rad(i)^2*eye(2);
    if obst.rad(i)==0
        grad(:,i)=[0;0];
    else
        grad(:,i) = mu/((r-r_obs)'*P_inv*(r-r_obs)-1)*(eye(2) -...
            P_inv*(r-r_obs)*r'/((r-r_obs)'*P_inv*(r-r_obs)-1))*Q*r;
    end
end
grad = sum(grad,2);
end
```

Listing B.10: Receding Horizon Constraint File for Motorboat

```
function [g,h] = ...
    rh_thrust_con(z,N,A,B,uA,uB,Umax,W_max,Beta,sbar,obst,h,D)

Umax = 0;
dx = z(1:N);
dy = z(N+1:2*N);
tf = z(end);

x(1) = A(1);
y(1) = A(2);
u_sw(1) = uA(1);
v_sw(1) = uA(2);
dv = [dx;dy];
dT = tf/N;

for i=1:D

Dy = 0; % for now

u_sw(i+1) = u_sw(i)+dv(1,i);
v_sw(i+1) = v_sw(i)+dv(2,i);

x(i+1) = ...
    x(i)+dT*(u_sw(i)+(dT/2)*dv(1,i)/dT)+(4*Umax/(h^2))*(dT^4*...
         ((Dy*v_sw(i))/4 -(dv(2,i)/dT*v_sw(i))/4)+...
          h*(((dv(2,i)/dT)/6-Dy/6)*dT^3+(v_sw(i)*dT^2)/2+y(i)*dT)...
         -dT^3*(v_sw(i)^2/3-(Dy*y(i))/3+(dv(2,i)/dT*y(i))/3)-...
          dT*y(i)^2-dT^5*((dv(2,i)^2/dT^2)/20 ...
         -(Dy*dv(2,i)/dT)/10+Dy^2/20)-dT^2*v_sw(i)*y(i));
y(i+1) = y(i) + dT*(v_sw(i)+(dT/2)*(dv(2,i)/dT-Dy));
end

% Inequality constraints
g = [];
```

```
34 % Equality constraints
   h=[ y(end)-B(2);% Must finish at point B
       x(end)-B(1);
       u_sw(end)-uB(1);
       v_sw(end)-uB(2)];
```

References

1. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotic Reseach*, vol. 5, p. 60, 1986.
2. I. Lopez and C. R. McInnes, "Autonomous rendezvous using artificial potential function guidance," *Journal of Guidance, Control, and Dynamics*, vol. 18, 1995.
3. J. D. Munoz, *Rapid Path-planning Algorithms for Autonomous Proximity Operations of Satellites*. PhD thesis, University of Florida, 2011.
4. J. C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
5. S. Waydo, "Vehicle motion planning using stream functions," tech. rep., California Institute of Technology, 2003.
6. U. Ahsun and D. W. Miller, *Dynamics and Control of Electromagnetic Satellite Formations*. PhD thesis, Massachusetts Institute of Technology, 2007.
7. P. Vadakkepat, K. Chen Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proceedings of the 2000 Congress on Evolutionary Computation, 2000.*, vol. 1, IEEE, 2000.
8. A. J. Healey, "Guidance laws, obstacle avoidance, artificial potential functions," *IEEE Control Series*, 2006.
9. G. A. Boyarko, "System level simulation of artificial potential function guidance for a neutrally buoyant autonomous vehicle equipped with non-ideal actuators," in *AIAA Aerospace Sciences Meeting*, 2009.
10. A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. K. Hedrick, "An overview of the emerging field of cooperative uav control," in *43rd IEEE Conference on Decision and Control*, 2004.
11. G. P. Roussos, G. Chaloulos, K. J. Kyriakopoulos, and J. Lygeros, "Control of multiple non-holonomic air vehicles under wind uncertainty using model predictive control and decentralized navigation functions," in *47th IEEE Conference on Decision and Control*, 2008.
12. J.-O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," in *IEEE Transactions on Robotics and Automation*, 1992.
13. S. Shimoda, Y. Kuroda, and K. Iagnemma, "Potential field navigation of high speed unmanned ground vehicleson uneven terrain," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005.
14. D. E. Koditschek and E. Rimon, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, 1992.

15. R. A. Fields, "Continuous control artificial potential function methods and optimal control," Master's thesis, Air Force Institute of Technology, 2014.

16. D. E. Kirk, *Optimal Control Theory: An Introduction (Dover Books on Electrical Engineering)*. Dover Publications, 2004.

17. A. E. Bryson, *Dynamic Optimization*. Addison Wesley Longman, 1999.

18. C. G. Henshaw, "A unification of aritficial potential function guidance and optimal trajectory planning," in *Proceedings of the 28th Annual AAS Rocky Mountain Guidance and Control Conference*, 2005.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE (DD–MM–YYYY)	2. REPORT TYPE	3. DATES COVERED *(From — To)*
27-03-2014	Master's Thesis	Sep 2012-Mar 2014

4. TITLE AND SUBTITLE
Performance Characterization, Development, and Application of Artificial Potential Function Guidance Methods

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Stickney, Heather M, Capt, USAF

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENY-14-M-44

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory, Space Vehicles
Attn: Dr. Josué D. Muñoz
3550 Aberdeen Ave SE
Kirtland AFB NM 87117-5776
josue.munoz@kirtland.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RV

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution A. Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
The primary objective was to examine artificial potential function (APF) guidance performance when applied to systems with limited control authority in a dynamic environment and develop a hybrid guidance to improve algorithm convergence and computational cost. Performance with respect to both computation time and cost was improved by hybridizing the APF approach with receding horizon planning. Results showed that for the hybrid algorithm, computation time was improved from the optimal control solution while improving the convergence and cost from the APF solution. While the hybrid method greatly improved performance for a saturated system in dynamic environment, this was limited to a fully actuated system. When applied with indirect control, performance was improved, but did not converge. Based on this initial data, the hybrid approach shows promise in regard to implementation within a real-time guidance scheme, however, there is still work to be done before it will be fully effective. The secondary objective was to determine what classes of problems are well-suited to APFs or APF-hybrids. The data suggests that APFs and the hybrid algorithm proposed are best applied to fully actuated systems. Additionally, if external dynamics or substantial saturation exist, APF guidance performs better when supplemented with an alternative method.

15. SUBJECT TERMS
Artificial potential function, optimal control, receding horizon planning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	UU	119	Lt. Col. Jeremy S. Agte (ENY)
					19b. TELEPHONE NUMBER (Include Area Code) (937) 255-3636 x4667 jeremy.agte@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18